OPAMP TECHNICAL BOOKS
1033 North Sycamore Avenue
Los Angeles, CA 90038
Tel. Number (213) 464-4322
Mon -Saturday 9:30 - 5:30

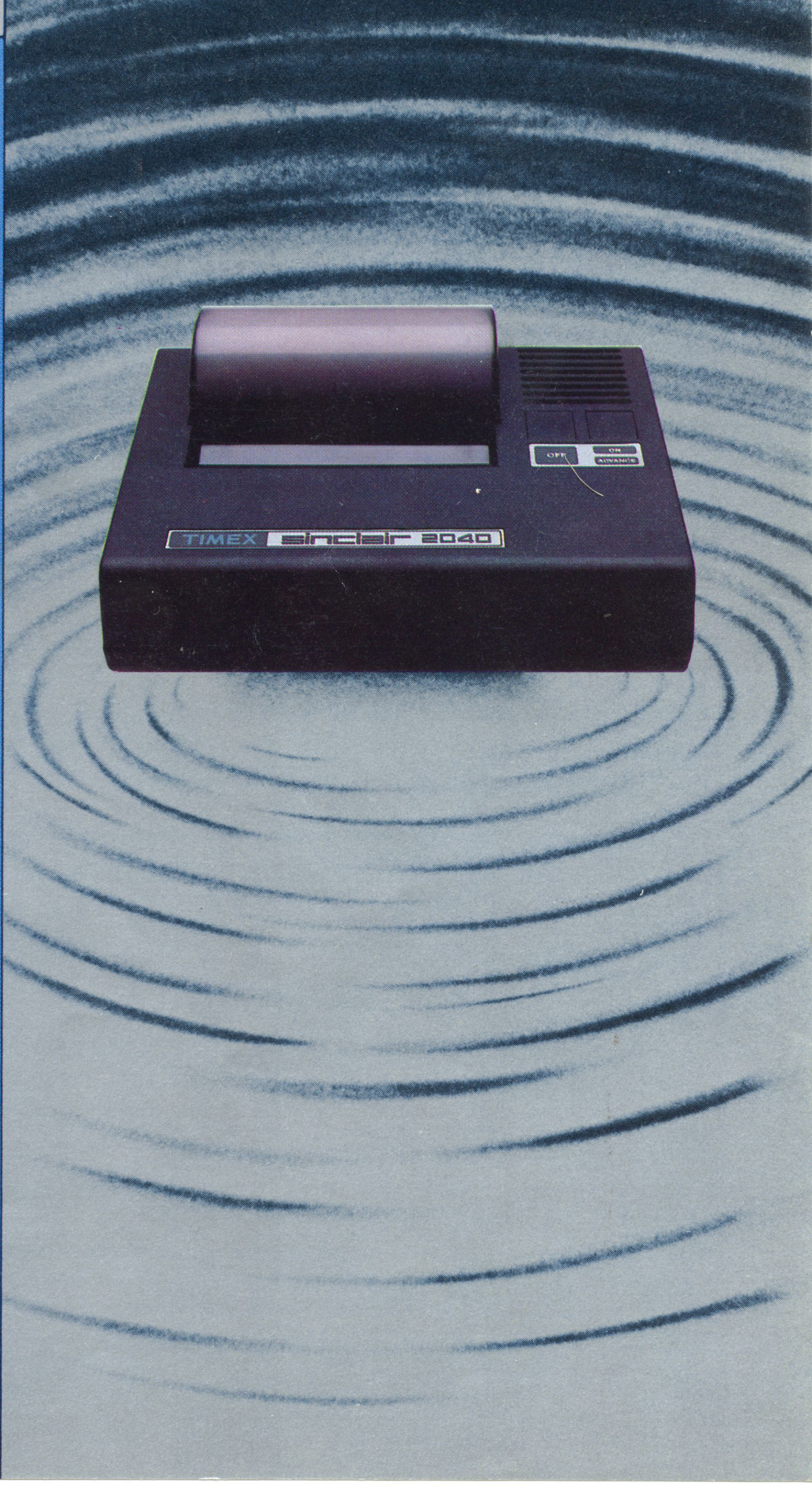
TIMEX PRINTER!

A PUBLICATION OF
THE HARVARD GROUP

SQ SYNTAX QUARTERLY

Serving The Timex-Sinclair Family of Personal Computers

Summer 1983 Vol. 2, No. 2



Exclusive Timex Printer Review

Build a 64K Dynamic RAM

Business Programs:
Income Tax Planner
Bond Yield

Moving It in BASIC *and* Machine Code

Beginners: Inside FOR-NEXT Loops

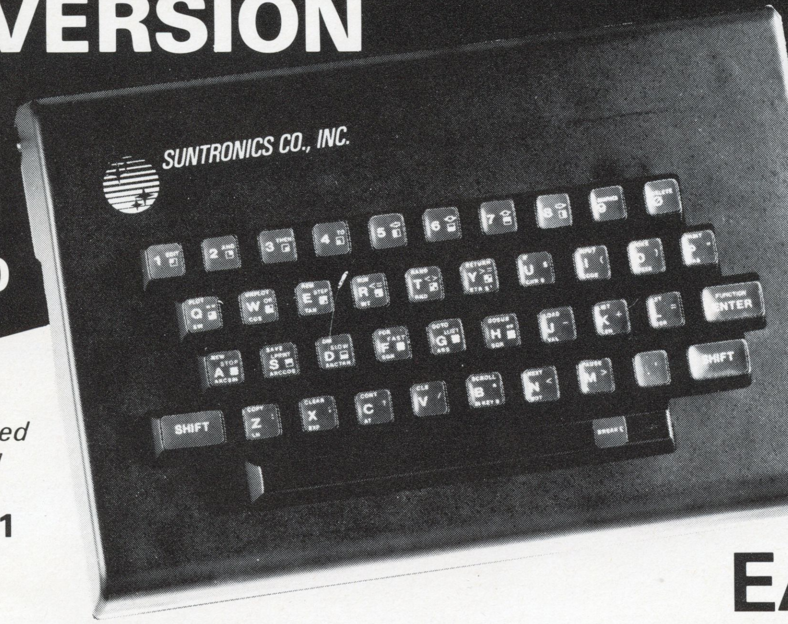
Sinclair's New 16K RAM Schematic

Plus Hardware and Software Reviews

0 26632 12083 05

FULL SIZE KEYBOARD CONVERSION

FOR YOUR ZX-81/ TS-1000



Only $69.95

Fully Warranteed For 90 Days!

SUN KD-81

EASY TO INSTALL

- **No Soldering**
- **No Modifications**

If you're tired of not knowing whether your data got entered or tired of poking data in with one or two fingers, then it's time to upgrade your ZX-81 to a full size, professional keyboard. The **SUN KD-81 KEYBOARD** offers all the ease and comfort of imputting your programs and text on a fast and efficient professional-sized keyboard!

KD-81 FEATURES

- **Full size keyboard with 41 keys**
- **Two color silk-screened key tops for easy reading**
- **Key tops have commands and graphics spelled out the same as ZX-81**
- **Extra shift key for real keyboard-style typing**
- **Full size space bar**
- **Allows touch typing**
- **Rear cutout allows any RAM or expansion module to be plugged in**
- **Keyboard comes with own cables**
- **Keyboard case holds both keyboard and computer with room to spare**
- **High impact plastic case with vaporized metal shielding**
- **Easy assembly — no soldering, no modifications**
- **Measures 10¾" × 7¼" × 2⅛"**

Check out these simple installation steps!

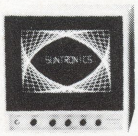
1. Remove 4 screws securing ZX-81 case and remove 2 screws holding ZX-81 PCB.
2. Unplug the 2 ribbon cables from the ZX-81 keyboard.
3. Plug the 2 ribbon cables into the connectors on the KD-81 keyboard.
4. Attach the ZX-81 PCB to the KD-81 case with 2 screws, close case and install remaining 4 screws into the bottom of the KD-81 Keyboard case ...

And enjoy the comfort and ease of inputting your data on a full size keyboard!

GET A BETTER PICTURE WITH SAMWOO MONITORS

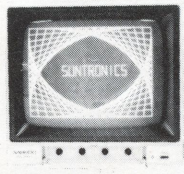
FOR YOUR SINCLAIR COMPUTERS

The SAMWOO provides a much better picture than your TV monitor. Includes easy-to-follow instructions for a simple modification.



9" 18Mhz BANDWIDTH

Green $121.00
Orange 125.00



12" 12Mhz BANDWIDTH

Green $130.00
Orange 134.00

Add $7.50 Shipping and Handling for this item.

Features:

- Composite Video Input/Output
- Switchable Input Impedance 75 or 10K ohm
- 750 Line Resolution at Center and 500 Lines at Corners
- Dimensions are 12.13" × 11.34" × 11.65" for the 12" model and 8.66" × 8.54" × 9.05" for the 9" model

SUNTRONICS CO., INC.

12621 Crenshaw Blvd., Hawthorne, CA 90250

STORE HOURS: MON.-FRI. 9:00 am to 6:00 pm
SATURDAY 10:00 am to 5:00 pm

CALIFORNIA
213-644-1149
(for Tech Info and Calif. orders)

OUTSIDE CALIFORNIA TOLL FREE
1-800-421-5775
(Order Desk Only)

Mail Order — Minimum Order $10. Send Money Order or Check to **P.O. BOX 1957 — DEPT. B, HAWTHORNE, CA 90250.** VISA or Mastercard (please include expiration date). Add $4.00 postage and handling to order. CA residents add 6% sales tax.

TIMEX PRINTER!

A PUBLICATION OF
THE HARVARD GROUP

SQ SYNTAX QUARTERLY

Serving The Timex-Sinclair Family of Personal Computers

Summer 1983 Vol. 2, No. 2

SQ Staff

Publisher
Kirtland H. Olson, P.E.

Editor
Ann L. Zevnik

Assistant Editor
Pamela Petrakos-Wilson

Editorial Assistant
Pamela K. Vinal

Advertising
Karen Brody

Production
CSA Press

Circulation
Constance Saarinen
Timothy Parker
Laura Rizzari
Todd M. Zevnik
Gail D. Olson

Financial Manager
Mary Russo

Bookkeeper
Marylin Morgan

Cover Photography
James Rue Design

SQ. Syntax Quarterly (ISSN 0734-0133) is copyright © 1983 and published by Syntax ZX80, Inc., a wholly owned subsidiary of The Harvard Group, RD 2, Box 457, Bolton Rd., Harvard, MA 01451. Subscription prices are $15 for one year (4 issues), single copies are $4.95. Third-class bulk rate postage paid at Harvard, MA 01451 and additional mailing offices. POSTMASTER: Send address corrections to The Harvard Group, RD 2, Box 457, Harvard, MA 01451.

Contents

Departments

SQ-ups

Keyboard Conversion by Dave Straub
In Winter SQ, p. 28, Figure 3, the numbers labelling the connectors should read in this order:

1 13 2 12 3 and 11 4 10 5 9 6 8 7

Thanks to Tim Osterhout and Bob Perrault for pointing out this correction.

Simultaneous Linear Equations and Matrix Inversion by R.A. Woodall
Mr. Woodall reported that in the third paragraph, p. 24 of the Spring SQ, the matrix dimensions 25x35 should read 35x35. A non-square matrix cannot be inverted.

Build Your Own EPROM Programmer and Centronics Printer Interface by John Oliger
Mr. Oliger sent the improved ROM-decoding circuit below. The old circuit did not include MREQ NOT. If you decode the ROM without MREQ NOT, the ZX printer will not work. In Part 1, Winter SQ, schematic 2 on p. 37, remove the resistor between pin 5 of U5 and Vcc. John originally added it to clean up the waveform, but since found that the TS1000's ROM CS NOT signal cannot pull this line to a good low logic level with this resistor installed. If the ROM CS NOT signal comes from one of the EPROM read boards, then it causes no problems.

On the PC layout for the EPROM programmer, p. 39, move the donut pad in the lower right corner of the component side 0.1 inch further to the *right*.

In Part 2, Spring SQ, cut off pin 19 of IC socket U4 before installing the socket on the EPROM programmer board. Pin 19 is not used and the ground trace runs under this pin, taking up space the pin's pad would use.

John adds that his price for bare boards, listed in Spring SQ, includes first class shipping within the US.

Program Improvements

Exploring String Functions by James A. Conrad
Robert Hartung writes: May I suggest the following addition to James Conrad's excellent presentation—

For printouts of columns of consecutive numbers, such as listing memory locations with their respective contents, you may want to print only the final digit(s). This truncation conserves display space and eases justifying and reading the columns.

It would seem this routine should work:

```
10 LET A=10001
50 PRINT A;
70 PRINT TAB 12;STR$ A((LEN ST
R$ A)-1 TO )
```

However, when we try to enter line 70 into the listing, we get a syntax error. (To avoid the error, enclose the first (STR$ A) in line 70 in parentheses so the computer creates the string from A before trying to slice it.—Ed.)

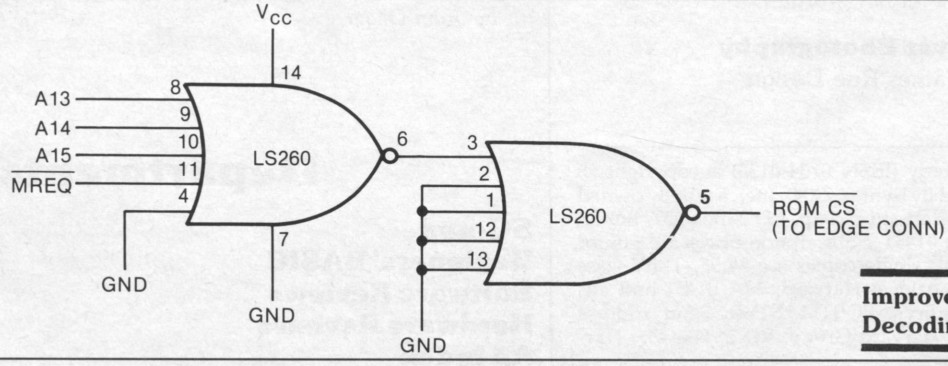
To solve the problem, key EDIT ENTER to delete line 70. Enter lines 20 and 60 following:

```
20 LET A$=STR$ A
60 PRINT TAB 12;A$(LEN A$)
```

When we run this revised version, we get the final character of A$. For a demonstration routine, enter:

```
  1 PRINT "    A";TAB 12;"A$";TAB
 20;"B$";TAB 28;"C$"
  5 PRINT
 10 LET A=10001
 20 LET A$=STR$ A
 30 LET B$=STR$ INT (A+0.5)
 40 LET C$=STR$ (INT (A*10+0.5)
/10)
 50 PRINT A;
 60 PRINT TAB 12;A$(LEN A$);
 70 PRINT TAB 20;B$(LEN B$-1 TO
);
 80 PRINT TAB 28;C$(LEN C$-2 TO
)
 90 LET A=A+1
100 REM LET A=A+1.1
110 GOTO 20
SYNTACTIC SUM: 18234, 8K ROM
```

Note that the REM in line 100 is a convenient way to deactivate a line without deleting it when testing the effects of various statements in a routine. RUN this demo, then delete the REM in line 100 and insert a REM after the line number and before the LET in line 90 to see the effect of increments of A by integral and fractional values. The 0.5 in lines 30 and 40 provides INT round-off to the nearest INT (integer). **SQ**



Improved ROM Decoding Circuit

Inside FOR-NEXT Loops

by James A. Conrad, Seattle, WA

One of your computer's many powerful features is its ability to perform a number of repetitive tasks. Doing a cyclical task over and over again is called looping in computer terms.

Beginning programmers often have difficulty understanding FOR-NEXT loops. When a computer carries out a loop, several things happen simultaneously. You must understand everything that's going on in the loop.

FOR Statements

All FOR statements have this general form:
FOR *Counter Variable* = *Initial Value* TO *Final Value* (STEP *Increment*)

Let's start with a simple example program:

```
10 FOR X=1 TO 5
20 PRINT X
30 NEXT X
```

(See the accompanying table for keyword locations on your ZX/TS keyboard.) RUN the program. You see this on your screen:

1
2
3
4
5

What's happening? The variable X in line 10 is a *counter variable,* sometimes called an index variable. It counts the number of times the loop executes. The FOR statement assigns the *initial value* (1 in this example) to the counter when the computer first enters the loop.

Line 20 is the "do" portion of the loop—one or several statements defining the process the computer performs. In this example it prints the value of the counter X.

Our NEXT statement in line 30 increments (adds 1 to) the counter. It then tests the counter against the *final value* to see if the loop has executed enough times. If it has not, the program loops back to line 20 with a new value for X and runs again.

Now let's make a few changes in our program and see what happens:

```
10 FOR K=2 TO 10 STEP 2
20 PRINT K
30 NEXT K
```

We changed the counter variable to K (you can use any single letter, non-subscripted, numeric variable). We also changed the initial value to 2 (it doesn't have to be 1) and added STEP 2. Now our program prints 2, 4, 6, 8 and 10, again in a column down the left of your screen.

STEP Function

STEP tells the NEXT statement how much to *increment* the counter. The increment doesn't have to be an integer (whole number)—change STEP in line 10 to STEP 0.5. Now our program prints 2, 2.5, 3, 3.5 and so on to 10.

If you want to count backward, just use a negative STEP. Change line 10 to read:

```
10 FOR K=5 TO 1 STEP -1
```

RUN it again. This time you see 5, 4, 3, 2 and 1 in a column on your screen.

If you leave out STEP altogether, the computer uses a default increment of 1, as in our first example.

Here's a program that tests various inputs. It uses variables for the initial value, final value and STEP increment (you could use expressions in place of variables):

```
10 PRINT "INITIAL VALUE ";
20 INPUT A
25 PRINT A
30 PRINT "FINAL VALUE ";
40 INPUT B
45 PRINT B
50 PRINT "STEP INCREMENT ";
60 INPUT C
65 PRINT C
70 FOR X=A TO B STEP C
80 PRINT X
90 NEXT X
```

RUN this program and experiment with various inputs. Try inputting an initial value of 1, a final value of 5 and a STEP increment of -1. Then try inputting an initial value of 7, a final value of 2 and a positive STEP increment. Logically these inputs shouldn't work and they don't on your Timex Sinclair. The first set tries to count backward from 1 to 5. The second attempts to count forward from 7 to 2. (Only designers of the federal budget can make this kind of logic work.) If you try this on other computers, however, you'll find many that will print the initial value in each of these cases. Their BASIC interpreters run through the loop once even if the logic is wrong—something to be aware of if you're debugging a program on another computer.

NEXT Statements

Some beginners find the NEXT part of FOR-NEXT loops difficult. The NEXT statement is where the computer increments the counter variable, adding the value of STEP to it, or subtracting if STEP is negative. It then compares the new value of the counter variable to the final value. If the new counter value is greater than the final value (or less if STEP is negative), the program continues to the line following the NEXT statement. If not, the program returns to the line following the FOR statement and goes through the loop again. The tricky part here is that when the computer completes the loop, the value of the counter variable is always *greater* (less if STEP is negative) than the final value.

Add this line to our last program:

```
100 PRINT "VALUE OF X IS ";X
```

RUN the program a few times until you're comfortable with the fact that the last counter value differs from the final value assigned by the FOR statement.

Before we move on to nested loops, study Table 1, which summarizes the workings of the parts of FOR-NEXT loops. When you understand how all the parts work, these loops will lose a lot of their mystery.

Inside Nested Loops

Nested FOR-NEXT loops consist of one loop inside another. They're easy to understand if you keep track of each counter variable's value at each step of the program execution. We'll start with a simple example:

```
10 REM NESTED LOOPS
20 FOR X=1 TO 3
30 FOR Y=1 TO 4
40 PRINT "WHILE X IS ";X,"Y IS
";Y
50 NEXT Y
60 NEXT X
```

(Remember, the computer does not execute REM statements.) Can you see from the screen display that the Y loop (called the *inside* loop) in lines 30 and 50 executes four times for each single execution of the X (*outside*) loop? Trace the program's execution on a sheet of paper, keeping track of the value of the counter variables X and Y.

Line 20 initializes the value of X as 1. Line 30 then initializes counter Y as 1. Line 40 prints the messages in the quotation marks and the current values of X and Y. Line 50, NEXT Y, increments variable Y to 2 and tests if it exceeds its final value of 4 (from line 30). It doesn't, so program control returns to line 40, the line following the FOR Y... statement. Line 40 again prints the messages and the current values of X and Y, 1 and 2 respectively.

Our program continues looping from line 50 to line 40, incrementing Y in line 50 and printing the new value of Y in line 40. X remains equal to 1 while the Y loop executes. Finally Y reaches 5, is tested against its final value of 4, and program control "falls through" to line 60.

Line 60, NEXT X, increments X to 2, tests it against its final value of 3, and passes control to the line following the FOR X... statement. This is line 30, which initializes a new Y loop. While the value of X is 2 the Y loop executes four times, just as it did while X was 1.

When Y reaches 5, program control again "drops through." X increments to 3 and a new Y loop begins. It's pretty simple, isn't it? For each time the outside (X) loop operates once, the inside (Y) loop operates completely.

Now a quick test: When the program finishes running, what are the ending values of variables X and Y? If you didn't answer 4 and 5, then you'd better loop back to the beginning of this tutorial.

Working with Arrays

We mainly use nested loops to work with data in multiple-dimension arrays. (Don't worry if you don't know a lot about arrays. We'll diagram the one we'll use.)

We'll set up an array, which we can diagram as:

		Columns				
		1	2	3	4	5
Rows	1	11	12	13	14	15
	2	21	22	23	24	25
	3	31	32	33	34	35

Figure 1.

Each array box is called an element. The values in the array boxes represent the row number as the first digit and the column number as the second. So row 1, column 1 contains the number 11 and row 3, column 4 contains the number 34.

This routine puts these values in their appropriate array elements. We'll use the variable R to designate the row number and C for column number. If you have trouble understanding what's happening, just enter these lines—it's easier to see how the loops work when the computer prints out the array elements (starting at line 200).

```
 10 REM DIMENSION ARRAY A, 3 RO
WS X 5 COLUMNS
 20 DIM A(3,5)
100 REM ASSIGN VALUES TO ARRAY
110 FOR R=1 TO 3
120 FOR C=1 TO 5
130 LET A(R,C)=R*10+C
140 NEXT C
150 NEXT R
```

Reading the Array

We've put the values for the array elements into the computer's memory, stored in an array named A. This array is the one diagrammed in Figure 1. Let's write a routine to print these values:

```
200 REM PRINT VALUES ROW BY COL
UMN
210 FOR R=1 TO 3
220 FOR C=1 TO 5
230 PRINT A(R,C);
240 NEXT C
250 NEXT R
```

This module prints a solid row of numbers across the screen beginning with the number 11 from row 1, column 1; then 12 from row 1, column 2 and so on. The trailing semicolon in line 230 suppresses the carriage return so the numbers print immediately following each other.

Can you see what happens? Line 210 initializes the row counter, R, with a value of 1. Then line 220 sets the column counter, C, to 1. The contents of the parentheses following the name of the array, A, in line 230 designate the row and column to print. At the beginning of the first pass through the loop, R and C both equal 1, so array element A(1,1) prints. This element is row 1, column 1, or the number 11.

Our program proceeds to line 240 where the NEXT C statement increments the column counter, C, to 2. Remaining in the C loop, the computer returns to line 230. R still equals 1. Line 230 now prints 12, the value of row 1, column 2.

Once again at line 240 the NEXT C statement increments C to 3 and returns to line 230. It's important to understand throughout the program execution what the value of each variable is. C has just been incremented to 3. R is still 1. The program is in the C loop and will remain in it until the column counter, C, exceeds 5, its final value in the FOR statement in line 220.

Our program continues to loop between line 240 (which increments the column counter) and line 230 (which prints the value of the next column). After the value of row 1, column 5 (15) prints, line 240 increments C once again to 6. Line 240 then tests to see if C exceeds its final value of 5. Because 6 does exceed 5, the C loop is completed and program execution drops through to the next line.

Line 250, NEXT R, now increments variable R—the row counter—and loops back to line 220, the line following the FOR R... statement. The values of counters R and C at this point (before line 220 executes) are 2 and 6.

A New Outside Loop

Program control is now at line 220, FOR C = 1 TO 5. Do you remember what happens when a FOR statement executes? The computer resets the control variable (here the column counter, C) to the initial value (1 in this line). We are into a new C loop. While R equals 2, this C loop again counts from 1 to 5. Line 230 prints columns 1 through 5 of row 2. And line 240 increments variable C. At the end of the C loop, line 240 increments C to 6, tests it against the final value of 5, and falls through to the NEXT R statement in line 250.

Now is a good time to revise our program slightly. Add " "; (quote, space, quote, semicolon) to the end of line 230. This prints a space between the numbers. Add line 245 PRINT. What does this do? Why? Why position a PRINT statement between the C and R loops? Figure it out—trace the program's progress through the loops keeping track of the current values of variables R and C. Jot down on a piece of paper what line 230 will print on each pass. (Hint: line 245 "prints" a blank line, technically called a line feed and carriage return.)

Did you predict correctly what the program printed on the screen? If so, congratulations—you have mastered one of the most difficult topics for beginning programmers. Our next module should be easy.

Reversing the Nesting

Our final module reverses the looping, nesting the R loop inside the C loop. Again study the progression through the loops, keeping the contents of the control (counter) variables in mind (or better, in little boxes on a sheet of paper) at all times.

```
300 REM YOU ENTER WHAT THIS LOO
P WILL DO
310 FOR C=1 TO 5
320 FOR R=1 TO 3
330 PRINT A(R,C);" ";
340 NEXT R
345 PRINT
350 NEXT C
```

Note that although we reversed the order of nesting, the positions of variables R and C inside the parentheses in line 330 remain R first and C second. This is because we are using the convention of row as the first variable and column as the second. No matter how we change the loop nesting, the computer interprets the first variable as the row position and the second as the column position.

Programming Tips

1. Use the same variables as counters in all your programs. This makes your programs easy to analyze and debug. Many programmers (including myself) use the variables X, Y and Z. I also reserve variables R and C as row and column counters for arrays.

2. Use a different set of variables for counters in subroutines. As you become more experienced, you'll build a library of subroutines that you use frequently and plug into your programs automatically. I started using different counter variable in subroutines after I went from an X loop in the main program into an X loop in a subroutine and wondered why my program didn't work. I now use variables J, K and L for counters in subroutines.

3. Don't jump out of a FOR-NEXT loop. Once your computer enters a loop, it wants to complete it. To halt execution of a loop, use an IF-THEN GOTO statement to jump to the line immediately before the NEXT statement. Use the same IF-THEN condition in this line to set the value of the counter variable to it final value. The NEXT statement will execute properly. For example:

```
100 FOR X=1 TO 10
110 ...portion of routine to be
executed
140 IF (condition is true) THEN
GOTO 180
150 ...portion of routine to be
skipped
180 IF (condition is true) THEN
LET X=10
190 NEXT X
```

4. Don't jump into a FOR-NEXT loop (for example, by using a GOTO statement from another part of your program). The NEXT statement looks for a counter variable that has been initialized by a FOR statement. If it finds none because you jumped into the loop, skipping the FOR statement, you'll get an error code 1 or 2 and your program stops.

5. If you want to delay your program for a period of time (for example, to leave a message on the screen for

10 seconds), use a timing loop (sometimes called a delay loop). The form for this is:

```
linenumber FOR X=1 TO (delay)
linenumber NEXT X
```

The computer does nothing but count from 1 to whatever you set as the final value. The delay on your ZX/TS is about 60 loops per second, so for a 10-second delay, use 600 for your final value. When your timing loop is near the end of a long BASIC program, this delay is longer, so test for the correct final value of your loop.

Another, easier way to delay a program is to use the PAUSE n statement (n being the number of pauses at 60 per second). The PAUSE statement, however, isn't available in most BASIC dialects. So if you want to write a portable program you can use on other systems, use a timing loop.

Because people's reading speeds vary, a better way of leaving messages on the screen is to use an INPUT statement at the end of your message. For example, try:

```
linenumber PRINT ''PRESS ENTER TO CONTI
NUE''
linenumber INPUT R$
```

to allow users to read the message at their own speeds.

Table 1.

FOR-NEXT Summary

FOR Statement:

Form: FOR *Counter Variable*=*Initial Value* TO *Final Value* (STEP *Increment*)

Counter Variable:

1. Assigned Initial Value by the FOR statement when the computer first enters loop.
2. Must be a single letter numeric variable (not subscripted).
3. Incremented in NEXT statement by STEP value (default is 1).
4. Can be a constant, variable or expression.
5. Can be positive or negative.
6. Value can be changed within be changed within the loop.

Initial Value:

1. Follows equal sign in FOR statement.
2. Assigned to Counter when computer first executed FOR statement.
3. Can be a constant, variable or expression.
4. Can be positive or negative.
5. Value cannot be changed from within the loop.

Final Value:

1. Follows TO in FOR statement.
2. Used to determine if Counter exceeds limit after being incremented.
3. Value set when computer first executes FOR statement.
4. Can be a constant, variable or expression.
5. Can be positive or negative.
6. Value cannot be changed from within the loop.

STEP:

1. NEXT statement increments Counter by the value of STEP.
2. Optional—computer uses value of 1 if you do not include STEP in FOR statement (default value).
3. Can be a constant, variable or expression.
4. Can be positive or negative integer or decimal.
5. Value cannot be changed from within the loop.

NEXT Statement

Form: NEXT *Counter Variable*

1. Increments Counter by value of STEP; defaults to 1 if you omit STEP.
2. Checks the incremented value of the Counter against the Final Value; if the Counter value exceeds (is less than if STEP is negative) the Final Value the program proceeds to the statement following NEXT. This is called falling or dropping through the loop.
3. You must state the Counter Variable after NEXT on your ZX/TS system (it is optional in many BASIC dialects).
4. In a nested loop the first NEXT statement must refer to the Counter Variable in the last FOR statement.

Nested Loop:

1. Means a loop inside another loop.
2. Each time the outside loop executes once, the inside loop executes completely.
3. Each FOR statement must have its own NEXT statement. The first NEXT must apply to the last FOR.

Where the Keys Are

If you're not accustomed to your ZX/TS's keyboard, use the following chart to find the keys used in the accompanying article on FOR-NEXT loops.

Statement/Operator	Keyboard Position
FOR	Above F
TO	Shift 4
NEXT	Above N
STEP	Shift E
LET	Above L
PRINT	Above P
INPUT	Above I
PAUSE	Above M
=	Shift L
(	Shift I
)	Shift O
;	Shift X
,	Shift .(period)

SQ

TRANSLATION TABLE
String Functions

No "standard" BASIC language exists — only dialects. This table summarizes the main differences between most common microcomputer BASIC dialects and Sinclair BASIC (used in ZX81s and Timex Sinclair 1000s). The table refers only to statements and operations used in or related to the accompanying article. It should help you translate programs written in common dialects into Sinclair BASIC.

Common BASICs	What It Does or Is	How To Use It in Common BASIC	Sinclair BASIC	Translation
Counter Variable FOR XK = . . .	'Index' or 'control' variable. Establishes loop to be executed and counts number of executions.	Any valid, non-subscripted, numeric variable.	Same — but single letter variable only.	Use single letter counter variables e.g. FOR X. . .
NEXT Y,X, . . .	Single NEXT statement with list of counter variables. Used in nested loops.	Many dialects allow.	Not available.	Write NEXT statements on separate lines — e.g. NEXT Y NEXT X
NEXT	NEXT statement with counter variable not named. Applies to last FOR statement.	Most dialects allow.	Not available.	Add name of counter variable.
PAUSE n	Halts program execution n frames (about 60/sec) or until any key is pressed.	Most dialects don't use.	Available.	Use delay loop (Tip 5) for portability.
Jumping out of a loop.	Transfer of program control out of a loop with GOTO statement. Bad programming practice.	Most dialects don't like; some don't allow.	No restrictions.	Avoid. See Tip 3 for method.
NF Error	NEXT without FOR error . . . Usually generated by: 1) Jumping into a loop or not writing FOR statement.	Most dialects use.	Report code 1 or 2.	Don't jump into a loop. Be sure there is a NEXT for each FOR.
	2) Improper placement of counter variable in nested loop NEXT statements (first NEXT must use last FOR counter).	Most dialects use.	No error report. Program executes but with different results.	Be certain first NEXT statement uses last FOR counter.
Colon (:) IF. . . THEN . . . : . . .	Statement separator. Allows two or more statements on one numbered line. Often used with IF . . . THEN statements in FOR . . . NEXT loops.	Most dialects allow.	Only one statement permitted per line.	Put each statement on separate line. (May require logic rearrangement with IF. . .THEN statements.

Got a Great Program?

Or a keen grasp of a ZX/TS concept? Would you like to see your name in print? **SQ** and **SYNTAX** newsletter are looking for authors. If you have a program or article you'd like to share with other ZX/TS users, send it along.

PROGRAMS: most programs are candidates for **SYNTAX** or **SQ**. We publish game, business, educational, utility and science/math programs. Please send programs on tape. Include as much documentation as you can, including a list of variables and their functions. Write complete instructions for using your program so the most inexperienced user could not mess it up. Sample output also helps. Make your program as user-friendly as possible within memory limitations. Indicate how much memory your program uses. And give the Syntactic Sum of the program as listed. For a free copy of Syntactic Sum, write SYNTAX.

ARTICLES: Articles can cover virtually any topic related to ZX/TS computers or applications. Our readers range from expert to novice, so we look for articles at all levels of complexity. If you present ideas on programming, it helps to include sample routines. If your article discusses hardware, please include sketches, schematics, or photos.

For both programs and articles, type your submission or print clearly, double-spaced. Include your name, address and day-time phone number. If you want your material returned, send an SASE.

What's in it for you? Aside from the fame and glory of appearing in **SYNTAX** or **SQ**, you'll receive a check. Programs receive a token $2 payment each. We pay 7 cents for every 6 published characters, including spaces and punctuation. Not enough to retire on, but often enough to pay for your subscription. This payment buys us all nonexclusive rights to your material. We can use it and so can you.

So share! Address submissions to:

Editor: SYNTAX/SQ
Rd 2, Box 457, Harvard, MA 01451 USA

MODULES FOR:

TIMEX-Sinclair

MODEM $119.95 KIT (RS-232) Port & Software Included)

$149.95 ASSEMBLED

64-K MEMORY $129.95 KIT

the "ULTIMATE MEMORY"

$139.95 ASSEMBLED

FEATURES:

- Battery backup
- PROM/ROM socket
- Reset Switch
- **BYTE-BACK EXCLUSIVE FULL 64-K**

New

The O-8K area is available. You can execute a copy routine (provided) to copy the TIMEX ROM into the 0-8K area of RAM then flip a switch and you have your operating system in RAM. You can modify it and create your own customized operating system.

WHY PAY MORE FOR LESS FEATURES?
GET THE "ULTIMATE MEMORY" BYTE-BACK UM-64

A COMPANY YOU CAN DEPEND ON

BYTE-BACK CO. is the oldest and one of the world's largest suppliers of modules for the ZX81 and TIMEX-Sinclair 1000 computers. There are thousands of BYTE-BACK modules in use with a track record of proven reliability.

RS-232 PRINTER INTERFACE Only $59.95 KIT $69.95 ASSEMBLED

CONTROL MODULE 8 Relays, 8 Inputs, $59.95 KIT $69.95 ASSEMBLED

16K MEMORY UM16 $59.95 KIT $69.95 ASSEMBLED & TESTED.
Battery backup, reset switch, PROM/ROM socket PLUS. . . .
1 year, 100% trade-in credit towards the **UM-64.**

BYTE BACK CO.
Rt. 3, Box 147, Brodie Road
Leesville, S.C. 29070
ORDER PHONE 803-532-5812
Add $4.95 shipping & handling to all orders.

3

90 Day Warranty On All Modules. 10—Day Return Priviledge

***SINWARE** provides these quality machine code programs for the ZX81/TS1000:*

STEP

STEP provides line-at-a-time execution of your BASIC programs and shows you the display and variable values to pinpoint programming errors. Set line breakpoints, loop breakpoints, or conditional breaks for fast testing of long routines. **STEP** occupies 3K at the top of your 16K RAM as you write and test new programs. Detailed documentation. The final solution for BASIC bugs.

Z-TOOLS

Z-TOOLS lets you merge programs from tape, renumber lines (including GOTOs and GOSUBs) for neat listings, copy and delete program blocks for quick restructuring, or verify tape contents against memory to eliminate program losses. Supplied in two versions, for the top of a 16K RAM pack or for the 8-10K block of expanded RAM.

HOT Z

HOT Z disassembles, debugs, and lets you copy and edit machine code programs. If you can move a cursor around and understand hex numbers, you can use HOT Z almost at once. Provides beautiful assembly listings, addressable cassette functions, runs your ZX printer (or Memotech I/F), even disassembles the ZX floating-point language. You owe your ZX/TX a fresh dose of **HOT Z.**

Z EXTRA

Z EXTRA is a display manager and data filer that lets you enter text, data or pictures directly to the screen and save them in groups in memory, on tape or in print. Display them sequentially or in scrolls through one another or use them in your BASIC programs. Give your computer a completely new personality with **Z EXTRA.**

PRICES

STEP	**Cassette**	**$14.95**
Z-TOOLS	**Cassette**	**$14.95**
Z EXTRA	**Cassette**	**$19.95**
HOT Z	**16K or 32K + Cassette**	**$19.95**
HOT Z-E	**Four 2716 EPROMs**	**$40.00**

SINWARE
Box 8032
Santa Fe, NM 87504

ZX81 & TS1000 Support!

Software/Hardware/Books from GLADSTONE ELECTRONICS

Best selling cassette software

ZX Forth

A complete implementation of the FORTH language for the ZX81 and TS1000 computer.
FORTH's most distinctive feature is its flexibility. The basic unit is the word — the programmer uses existing words to define his own which can then be used in further definitions. FORTH is a compiled language so programs run very fast (typically five times faster than BASIC).
ZX-FORTH is supplied on cassette and is accompanied by extensive documentation:
56-page Users Manual 8-page Editor Manual
Z43 **$29.95**

Travel the world with your T/S1000 - ZX81 Computer. Answer the 12 riddles and WIN! ZX48 **$19.95**

ZX Bug

Machine Code Monitor and Disassembler

ZXBUG is a powerful tool for machine language programming. It is 4K long and uses memory from 71EO to the top memory. ZX BUG works in hexadecimal (base 16), not decimal, so all addresses are a maximum of 4 Hex bits long. Provides a total of 28 commands. Z41 **$14.95**

ZX Assembler

This Machine Code program occupies 7K of memory and locates itself at the top of memory. The program is a full Editor/ Assembler and Monitor. Labels may be used instead of any string. The features include Line Insertion/Delete, Insert Characters, Auto Repeat on all keys. The Monitor has facilities to inspect memory, registers and run machine code programs. Z40 **$14.95**

New Software

Tiny Logo

Teach your children programming skills through a computer graphics language. It's easy and fun to use! Included with your program is a comprehensive guide. (Tiny Logo is a subset of the popular MIT Logo) **$19.95**

ZX TS fastload

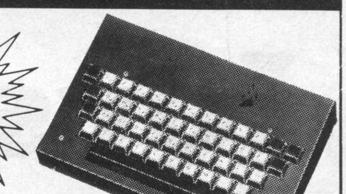
Fastload allows you to save and load programs at rates up to 6 times faster than normal. Change your old programs into new FASTLOAD programs and no extra equipment is needed. Z61 **$19.95**

The Complete ZX81 and TS1000 Library!

BOOKS

The Complete ZX81 Basic Course	
Basic Course Cassettes (2)	B70 $29.95
Getting Acquainted with your ZX81	C70 $8.95
ZX81 Pocket Book	P70 $10.95
Making the Most of Your ZX81	P71 $10.95
Computer for Kids, ZX81 Edition	C71 $4.95
Explorers Guide to the ZX81	T70 $12.95
ZX81 Companion	C72 $8.95
49 Exploding Games for the Zx81	P72 $10.95
Not Only 30 Programs for ZX81 : 1K	B72 $9.95
Machine Language Made Simple	B73 $14.95
Mastering Machine Code on your ZX81	P73 $12.95
ZX81 Rom Disassembly: Part A	Both for
ZX81 Rom Disassembly: Part B	$19.95
Understanding your ZX81 ROM	B76 $14.95

$99.95



KEYBOARD with metal case

A full -sized professional keyboard for the ZX81 Features 47 keys and a full-sized space bar. Connects to the ZX81 with no soldering required, via a plug-in flexible connector. You can purchase the keyboard only, or the optional metal case that holds both keyboard and ZX81. Expansion devices (i.e RAM packs, etc) connect to the ZX81 edge-connector which extends from the rear of the cabinet.
A professional keyboard makes program entry easier and less error-prone.
Keyboard K70 **$79.95**
Metal Case for keyboard and ZX81 E70 **$25.00**

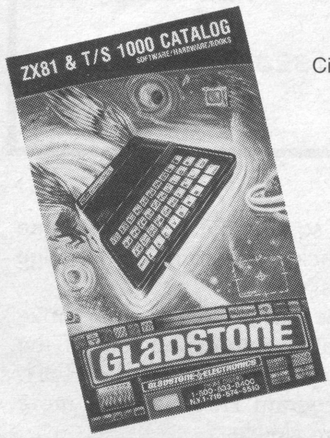
GLADSTONE 64K RAM

Expands the capacity of the T/S 1000 or ZX81 to its maximum. User transparent. G64 **$149.95**

Cramic-81 CMOS 16K Memory

Cramic-81 is a 16K RAM pack using CMOS technology for permanent data retention. Data can be stored almost indefinitely and loaded almost instantly with virtually zero chance of error. An ordinary Rampack can also be connected simultaneously. The two RAM units then occupy the same memory area but either can be selected. Two separate programs can actually run alternately. C12 **$179.95**

ORDER TOLL FREE
800-833-8400
In N.Y. (716) 874-5510
In Canada 1-800-268-3640 Monday-Friday 9:00 am. - 1:00 am E.S.T



Write for Gladstone's
FREE CATALOG
Circle reader service number or write
(no phone requests please!)

New from Gladstone Electronics! Our ZX81/TS1000 catalog will take you where no one has dared go before.! You will view the widest selection of up-to-date software, books and hardware add-ons available to get the most from your personal computer. This exciting new 34 page color catalog lists arcade, fantasy and family games, business and educational programs; books for beginners as well as experienced users; hardware add-ons and other peripherals for use with ZX81/TS1000 Home Computer! Use the convenient coupon and send for your FREE catalog TODAY!

GLADSTONE ELECTRONICS
1585 Kenmore Ave. Buffalo N.Y. 14217

ORDER FORM

CHARGE TO:
☐ VISA ☐ MASTERCARD
account number
Expiry date
Signature
Date
NAME
ADDRESS
CITY: STATE: ZIP:

Please rush me:

	Quantity	Price	Total

New York Res. add sales tax
SHIPPING
TOTAL

To: Gladstone Electronics, 1585 Kenmore Ave., Buffalo N.Y. 14217

In Canada: 1736 Avenue Rd., Toronto, Ontario M5M 3Y7

CANADA: PRICES MAY VARY

by Dan Tanberg, M.D., Calliope Software, Albuquerque, NM 8K ROM/2K RAM

Software Joystick in BASIC and MC

One problem with Timex Sinclair computers is that they don't come with a joystick control. New users may soon be frustrated by the relative slowness and awkwardness of many BASIC graphics game programs. While machine language programming can overcome these difficulties, many users have neither the time nor the inclination to learn it. Even so, a variety of tricks can help you speed up and improve the quality of your video games written in BASIC. We'll look at some way to improve keyboard control of your game programs. We will also introduce a machine code "joystick" subroutine that you can use in any program. Our final program, Mousemaze, requires 2.5K RAM, but the others fit in only 2K RAM.

Getting Your Graphics Around in BASIC

Program One exemplifies a primitive graphics game. Type it in and RUN it. You'll see that it PRINTS and then erases a black square while allowing you to move about the screen using the cursor keys: 5, 6, 7 and 8.

```
   1 REM PROGRAM ONE
  10 LET ROW=10
  20 LET COL=15
 100 PRINT AT ROW,COL;"■"
 120 PRINT AT ROW,COL;" "
 200 IF INKEY$="8" THEN LET COL=
COL+1
 210 IF INKEY$="5" THEN LET COL=
COL-1
 220 IF INKEY$="6" THEN LET ROW=
ROW+1
 230 IF INKEY$="7" THEN LET ROW=
ROW-1
 300 GOTO 100
SYNTACTIC SUM:   13219, 8K ROM
```

This program's major problem is that the cursor keys are located in an awkward position for coordinated, serious game-playing. Program Two corrects this by choosing more comfortable keys to move the black square about. Change lines 210, 220, and 230 as shown in Program Two and RUN it again. Now you can use your left middle and index fingers on E and F to move the square up and down. Use your right index and middle fingers on Y and 8 to move it left and right. Much more convenient, isn't it?

Some game players find that it helps to stick tiny (1 mm) pieces of cellophane tape on these keys so their fingers don't get lost. I also find this helpful.

```
   1 REM PROGRAM TWO
  10 LET ROW=10
  20 LET COL=15
 100 PRINT AT ROW,COL;"■"
 120 PRINT AT ROW,COL;" "
 200 IF INKEY$="8" THEN LET COL=
COL+1
 210 IF INKEY$="Y" THEN LET COL=
COL-1
 220 IF INKEY$="F" THEN LET ROW=
ROW+1
 230 IF INKEY$="E" THEN LET ROW=
ROW-1
 300 GOTO 100
SYNTACTIC SUM:   13288, 8K ROM
```

Now, if you haven't done so already, hold down one of your new keys until the black square runs off the edge of the screen. What happens?

Program Three corrects this flaw by limiting your move in any direction, then stopping before you fall off the edge. Change lines 200 through 230 as shown and RUN. See? Much better, isn't it? Program Three is much more "user friendly," but runs more slowly.

```
   1 REM PROGRAM THREE
  10 LET ROW=10
  20 LET COL=15
 100 PRINT AT ROW,COL;"█"
 120 PRINT AT ROW,COL;" "
 200 IF INKEY$="8" AND COL<31 TH
EN LET COL=COL+1
 210 IF INKEY$="Y" AND COL>0 THE
N LET COL=COL-1
 220 IF INKEY$="F" AND ROW<21 TH
EN LET ROW=ROW+1
 230 IF INKEY$="E" AND ROW>0 THE
N LET ROW=BOW-1
 300 GOTO 100
SYNTACTIC SUM:  16063, 8K ROM
```

Program Four demonstrates a way to move the square more smoothly. Just move line 100 down to 250 at the bottom of the loop and notice the change.

```
   1 REM PROGRAM FOUR
  10 LET ROW=10
  20 LET COL=15
 100 PRINT AT ROW,COL;" "
 200 IF INKEY$="8" AND COL<31 TH
EN LET COL=COL+1
 210 IF INKEY$="Y" AND COL>0 THE
N LET COL=COL-1
 220 IF INKEY$="F" AND ROW<21 TH
EN LET ROW=ROW+1
 230 IF INKEY$="E" AND ROW>0 THE
N LET ROW=ROW-1
 250 PRINT AT ROW,COL;"█"
 300 GOTO 100
SYNTACTIC SUM:  16175, 8K ROM
```

You can further smooth the cursor's motion by making the alterations shown in Program Five. Add two additional variables (XROW and XCOL) to remember the old PRINT row and column numbers so we can erase the old square just before printing the new one. Edit Program Four to create Program Five and try it. Remember to change line 300 to GOTO 80 and to move line 120 downward with its new variables XROW and XCOL.

```
   1 REM PROGRAM FIVE
  10 LET ROW=10
  20 LET COL=15
  80 LET XROW=ROW
  90 LET XCOL=COL
 200 IF INKEY$="8" AND COL<31 TH
EN LET COL=COL+1
 210 IF INKEY$="Y" AND COL>0 THE
N LET COL=COL-1
 220 IF INKEY$="F" AND ROW<21 TH
EN LET ROW=ROW+1
 230 IF INKEY$="E" AND ROW>0 THE
N LET ROW=ROW-1
 240 PRINT AT XROW,XCOL;" "
 250 PRINT AT ROW,COL;"█"
 300 GOTO 80
SYNTACTIC SUM:  18045, 8K ROM
```

Combining lines 240 and 250, as in Program Six, further improves the smoothness and speed. This change avoids the delay caused by making the operating system decode and execute two separate PRINT statements. Make the change and try it. You'll see a real difference.

```
   1 REM PROGRAM SIX
  10 LET ROW=10
  20 LET COL=15
  80 LET XROW=ROW
  90 LET XCOL=COL
 200 IF INKEY$="8" AND COL<31 TH
EN LET COL=COL+1
 210 IF INKEY$="Y" AND COL>0 THE
N LET COL=COL-1
 220 IF INKEY$="F" AND ROW<21 TH
EN LET ROW=ROW+1
 230 IF INKEY$="E" AND ROW>0 THE
N LET ROW=ROW-1
 240 PRINT AT XROW,XCOL;" ";AT R
OW,COL;"█"
 300 GOTO 80
SYNTACTIC SUM:  17428, 8K ROM
```


SPELLING COACH

STUDENTS . . .
PARENTS . . .

Studying for that weekly spelling/vocabulary test can be boring. With the **SPELLING COACH** it's fun. You type in the words you need to learn. The **SPELLING COACH** holds your attention and motivates you into taking practice tests until your spelling is perfect. The better your spelling gets, the more time you get to play the **SPELLING COACH's** own arcade game.

Requires ZX81/TS1000 with 16K. Send check or money order to:

B-C-L
360 Colwin Court
Blue Bell, PA 19422

Number of copies ______
@ 14.95 ______
Postage/handling 1.50
Total ______

NAME ______________________
ADDRESS ______________________
CITY/STATE/ZIP ______________________

Improving Speed

We improved our game considerably by making it more comfortable, more user friendly, and smoother, but we paid for these with a loss of speed. What can we do to regain some of this?

Program Seven shows one approach we can take. INKEY$ functions slowly and we used it four times each trip around the loop. By assigning a string variable (such as K$) to equal INKEY$ (as in line 150) we can get by with one INKEY$ per loop cycle. Add line 150 as shown in Program Seven and then change each INKEY$ to K$ in lines 200 through 230. Now RUN this and see the improved speed.

```
   1 REM PROGRAM SEVEN
  10 LET ROW=10
  20 LET COL=15
  80 LET XROW=ROW
  90 LET XCOL=COL
 150 LET K$=INKEY$
 200 IF K$="8" AND COL<31 THEN L
ET COL=COL+1
 210 IF K$="Y" AND COL>0 THEN LE
T COL=COL-1
 220 IF K$="F" AND ROW<21 THEN L
ET ROW=ROW+1
 230 IF K$="E" AND ROW>0 THEN LE
T ROW=ROW-1
 240 PRINT AT XROW,XCOL;" ";AT R
OW,COL;"█"
 300 GOTO 80
SYNTACTIC SUM:   18166, 8K ROM
```

Program Eight saves memory and speeds things up a little more by employing the logical operator AND to combine things. (Enter AND by using SHIFT 2.) In BASIC, logical statements such as K$ = "8" have a value of one if they happen to be true; and a value of zero if they're false. Two logical statements may be joined with an AND operator, for example, K$ = "8" AND X 1. The complete statement has a value of one *only* if both components are true; otherwise the value is zero.

Consider line 200 of Program Eight carefully. If you're pressing the 8 key, and the column number is also less than 31, then line 200 adds 1 to the variable COL. Can you see how this works?

Substitute lines 200 and 220 for the old key lines and RUN this. You can just perceive the speed increase.

```
   1 REM PROGRAM EIGHT
  10 LET ROW=10
  20 LET COL=15
  80 LET XROW=ROW
  90 LET XCOL=COL
 150 LET K$=INKEY$
 200 LET COL=COL+(K$="8" AND COL
<31)-(K$="Y" AND COL>0)
 220 LET ROW=ROW+(K$="F" AND ROW
<21)-(K$="E" AND ROW>0)
 240 PRINT AT XROW,XCOL;" ";AT R
OW,COL;"█"
 300 GOTO 80
SYNTACTIC SUM:   13398, 8K ROM
```

You can increase the game's speed more by avoiding INKEY$ altogether. Sinclair's operating system (that 8K machine language program living in your ROM) periodically scans the keyboard to see whether you're pressing a key. Each key has its own two-byte code number. When you push a key the computer stores that code number in addresses 16421 and 16422 (the systems variable LAST_K). You can PEEK either address and use the resulting code number in your games, much like using INKEY$. And it's faster.

Clear your machine by entering NEW. Then type in and RUN Program Nine. It waits for you to press a key, then prints that key, followed by the low byte of its own code number (we are looking at 16421, the address of the low byte of LAST—K). Try pressing different keys and see if you can figure out the pattern. (See Toni Baker's book, *Mastering Machine Code on your ZX81* from Reston Publishing Co., Reston, VA, 1981, pp. 88-89.) Now press our joystick keys: E, F, Y and 8. You'll get 251, 253, 223 and 239, respectively. Note these numbers—you'll use them later.

(Programs Nine and onward can respond in odd ways to multiple keypresses or use of keys other than E, F, Y and 8. If you use these modules, add error traps so that *only* the desired keys produce action.—Ed.)

```
   1 REM PROGRAM NINE
  10 IF INKEY$="" THEN GOTO 10
  20 LET K$=INKEY$
  30 LET A=PEEK 16421
  40 SCROLL
  50 PRINT K$,A
  60 GOTO 10
SYNTACTIC SUM:   5618, 8K ROM
```

Enter NEW again and turn to Program Ten. Lines 1-90 are familiar, but after that things get a little strange. Lines 230-560 contain five separate subroutines for moving the PRINT AT position left, right, up, down, or nowhere. REM statements appear at the start of each subroutine for clarity; omit these if you're short on RAM.

Line 100 is the confusing one. It PEEKs address 16421 to get the code number of the key you might be pressing. (If no key is pressed then PEEK 16421 = 255.) Line 100 then does some algebra to convert this value to the line number of the subroutine that creates the proper action. For example, if you press E, then PEEK 16421 equals 251; 251 minus 200 equals 51; 51 times 10 equals 510; and GOSUB 510 gets you to the subroutine to decrement the row number, ROW. RETURNing to the loop at line 110, we are now ready to PRINT. We've avoided translating many unnecessary lines of the program and have not used INKEY$ at all; but at the cost of additional memory. This tradeoff between speed and memory is always with us and only the very creative can beat it.

Enter Program Ten and press RUN; touch ENTER, but very quickly. You'll find this is noticeably faster than Program Eight. Can you do better?

```
   1 REM PROGRAM TEN
  10 LET ROW=10
  20 LET COL=15
  80 LET XROW=ROW
  90 LET XCOL=COL
 100 GOSUB ((PEEK 16421-200)*10)
 110 PRINT AT XROW,XCOL;" ";AT R
OW,COL;"■"
 120 GOTO 80
 230 REM ■■■ LEFT "Y"
 240 LET COL=COL-(COL>0)
 245 RETURN
 390 REM ■■■ RIGHT ("8")
 400 LET COL=COL+(COL<31)
 405 RETURN
 510 REM ■■■ UP ("E")
 520 LET ROW=ROW-(ROW>0)
 525 RETURN
 530 REM ■■■ DOWN ("F")
 540 LET ROW=ROW+(ROW<21)
 545 RETURN
 550 REM ■■■ NO KEY
 560 RETURN
SYNTACTIC SUM: 22752, 8K ROM
```

Program Eleven takes this idea one step further and moves a copy of each BASIC statement down into each "direction" subroutine. RUN Program Eleven in SLOW mode. It waits for your key press, then goes to the subroutine for that key. Not a bit of wasted time, but this method uses up more memory. Also observe how lines 244, 404, 524 and 544 hold you in the GOSUB loops until you remove your finger from the key you've pressed. Get the idea? Type in Program Eleven and RUN it, if you haven't already.

Touch one of the direction keys to start things moving. When you are not pressing any key, no flashing occurs. You can add some more lines to the "NO KEY" subroutine at line 550 if you want the black square to flash when it's not moving.

```
   1 REM PROGRAM ELEVEN
  10 LET ROW=10
  20 LET COL=15
 100 GOSUB ((PEEK 16421-200)*10)
 120 GOTO 100
 230 REM ■■■ LEFT "Y"
 232 LET XROW=ROW
 233 LET XCOL=COL
 240 LET COL=COL-(COL>0)
 242 PRINT AT XROW,XCOL;" ";AT R
OW,COL;"■"
 244 IF PEEK 16421=223 THEN GOTO
 232
 245 RETURN
 390 REM ■■■ RIGHT ("8")
 392 LET XROW=ROW
 393 LET XCOL=COL
 400 LET COL=COL+(COL<31)
 402 PRINT AT XROW,XCOL;" ";AT R
OW,COL;"■"
 404 IF PEEK 16421=239 THEN GOTO
 392
 405 RETURN
 510 REM ■■■ UP ("E")"
 512 LET XROW=ROW
 513 LET XCOL=COL
 520 LET ROW=ROW-(ROW>0)
 522 PRINT AT XROW,XCOL;" ";AT R
OW,COL;"■"
 524 IF PEEK 16421=251 THEN GOTO
 512
 525 RETURN
 530 REM ■■■ DOWN ("F")
 532 LET XROW=ROW
 533 LET XCOL=COL
 540 LET ROW=ROW+(ROW<21)
 542 PRINT AT XROW,XCOL;" ";AT R
OW,COL;"■"
 544 IF PEEK 16421=253 THEN GOTO
 532
 545 RETURN
 550 REM ■■■ NO KEY
 560 RETURN
SYNTACTIC SUM: 44167, 8K ROM
```

Program Twelve is about the best I can do in BASIC although I'm sure others can improve the speed a bit more. I simply moved all the lines in Program Eleven so the machine takes less time to serially search through all the line numbers to find the proper GOSUB. Changes to line 4 make the GOSUB line numbers come out right. Once you've got Program Twelve running properly, you can remove all those REM statements for a little additional speed.

```
   1 REM PROGRAM TWELVE
   2 LET ROW=10
   3 LET COL=15
   4 GOSUB ((PEEK 16421-221)*4)
   5 GOTO 4
   8 REM ■■■ LEFT "Y"
   9 LET XROW=ROW
  10 LET XCOL=COL
  11 LET COL=COL-(COL>0)
  12 PRINT AT XROW,XCOL;" ";AT R
OW,COL;"■"
  13 IF PEEK 16421=223 THEN GOTO
9
  14 RETURN
  72 REM ■■■ RIGHT ("8")
  73 LET XROW=ROW
  74 LET XCOL=COL
  75 LET COL=COL+(COL<31)
  76 PRINT AT XROW,XCOL;" ";AT R
OW,COL;"■"
  77 IF PEEK 16421=239 THEN GOTO
 73
  78 RETURN
 120 REM ■■■ UP ("E")"
 121 LET XROW=ROW
 122 LET XCOL=COL
 123 LET ROW=ROW-(ROW>0)
 124 PRINT AT XROW,XCOL;" ";AT R
OW,COL;"■"
 125 IF PEEK 16421=251 THEN GOTO
 121
 126 RETURN
 128 REM ■■■ DOWN ("F")
 129 LET XROW=ROW
 130 LET XCOL=COL
```

Continued Next Page

```
 131 LET ROW=ROW+(ROW<21)
 132 PRINT AT XROW,XCOL;" ";AT R
OW,COL;"█"
 133 IF PEEK 16421=253 THEN GOTO
 129
 134 RETURN
 136 REM ███ NO KEY
 137 RETURN
SYNTACTIC SUM:  43048, 8K ROM
```

A Machine Code "Joystick"

Program Twelve still leaves a lot to be desired. It is slower than you'd like for some applications, and you can't move diagonally by pressing two keys at once. To improve this I wrote an 82-byte machine language subroutine you can store in a REM statement and use to advantage in any graphics game programs you like. Even if you know nothing about machine code, you can still use this program effectively. Don't get scared off now that you've come this far; JOYSTICK is well worth having in your hands!

With JOYSTICK you'll enjoy all the features shown in BASIC programs One through Twelve preceding, plus increased speed and the ability to move diagonally by pressing two keys at the same time.

JOYSTICK uses our old friends E, F, Y and 8 keys to move around. If you hold down E and 8 together, the position moves up and to the right.

To get JOYSTICK's 82 bytes into a REM statement so you can use it, first enter the LOADER program. Line 1 of this program must contain at least 82 characters after the REM, though a few extra won't hurt anything. After you enter LOADER, proofread it carefully. Make sure that the semicolon in line 30 is there. SAVE this on tape.

Press RUN, then ENTER. Now put in the first number from the listing for JOYSTICK, decimal 10. ENTER it and a 10 should appear up on the screen just as in the listing. Cover up all but the working line of the decimal listing with a piece of paper so you don't get lost half way through the line. Enter machine code from left to right. When you've entered all 82 numbers, the screen looks just like the listing. After the last number, 201, goes in, the loader sums them all. This checksum equals 7627 if there are no errors. If it's not right, RUN the loader again and re-enter the 82 numbers correctly. When the checksum is OK, SAVE a copy on tape right away so you don't have to repeat all this. SAVE it under the name JOYSTICK. Better safe than sorry.

Loader

```
   1 REM 1234567890123456789012З
45678901234567890123456789012345
678901234567890123456789012
   2 DIM A$(82,4)
  10 FOR I=1 TO 82
  20 INPUT A$(I)
  30 PRINT A$(I);
  40 POKE 16513+I,VAL (A$(I))
  50 NEXT I
 100 REM CHECKSUM
 110 LET CHECKSUM=0
 120 FOR I=1 TO 82
 130 LET CHECKSUM=CHECKSUM+PEEK
(16513+I)
 140 NEXT I
 150 PRINT AT 18,0;"CHECKSUM= ";
CHECKSUM
 160 IF CHECKSUM<>7627 THEN PRIN
T "ERROR"
 170 IF CHECKSUM=7627 THEN PRINT
 "OK"
SYNTACTIC SUM:  21085, 8K ROM
```

JOYSTICK Machine Code Listing

10	15	10	15	58	130	64	50
132	64	58	131	64	50	133	64
205	187	2	125	254	255	200	203
77	32	11	58	130	64	254	21
40	19	60	50	130	64	203	85
32	11	58	130	64	254	0	40
4	61	50	130	64	203	109	32
11	58	131	64	254	0	40	4
61	50	131	64	203	101	192	58
131	64	254	31	200	60	50	131
64	201						

LIST the program and you'll see that the REM statement in line 1 contains the characters representing the machine language for JOYSTICK. Delete all the lines except the first, and SAVE this on tape, again as JOYSTICK. Make an extra for safety.

Next ENTER lines 10-30 of Program Thirteen. Take care to get the numbers and punctuation marks exactly right in line 20 or it won't work. Make sure your computer is in SLOW mode, RUN, then ENTER. You'll be able to move our old friend, the black square, around with E, F, Y and 8. Impressive speed, isn't it? (Almost all the time is spent decoding and executing BASIC lines 20 and 30, not the machine code JOYSTICK.)

Now press Y and F. See what happens? You now have control of all four diagonal directions and more speed. We've come a long way since Program One, haven't we?

```
   1 REM ▒7▒7U▄RNDM▀RNDU▄RNDM█RN
DLN █▀7 RETURN COPY COS ACS 74"U
▄RND RETURN +C<WM▄RNDACS 74"U▄RN
D RETURN  C▪XM▄RNDACS 74"U▄RND R
ETURN  C▪XM▄RNDACS ?""U▄RND RETU
RN 3COS WM▄RNDTAN
  10 RAND USR 16518
  20 PRINT AT PEEK 16516,PEEK 16
517;" ";AT PEEK 16514,PEEK 16515
;"█"
  30 GOTO 10
SYNTACTIC SUM:  13568, 8K ROM
```

You can use JOYSTICK in any program in which you wish to move something around the screen with keys. Use the REM statement containing the machine code that you saved on tape as the first line of your program. The first two bytes of the machine code are where we store the machine equivalent of our BASIC variables ROW and COL. So now instead of:

PRINT AT ROW,COL;" "

use

PRINT AT PEEK 16514,PEEK 16515;" "

And for

PRINT AT XROW,XCOL;" "

use

PRINT AT PEEK 16516,PEEK 16517;" "

For more speed, combine these into one PRINT statement as in line 20 of Program Thirteen.

Call JOYSTICK with USR 16518. Just include a line such as RAND USR 16518 in your main loop, usually right before your PRINT statement.

To initialize the position where you want your object to start out, POKE the row number into address 16514 and the column number into 16515 early in your program. See lines 6020 and 6030 of the following sample program, Mousemaze, for an example of how to do this.

If you need boundaries for your program other than rows 0 and 21 and columns 0 and 31, here's how to change them. First LOAD the REM statement containing JOYSTICK.

Then POKE in the limits you need:

POKE 16545,bottom row number
POKE 16560,top row number
POKE 16575,left column number
POKE 16589,right column number

Now your moving object stays within the newly set bounds.

An Example—Mousemaze

Mousemaze shows how you can effectively use JOYSTICK in a typical BASIC maze program. It uses just under 2.5K RAM. You use the E, F, Y and 8 keys to steer your hungry mouse, represented as an asterisk, toward the cheese, shown as C, in the shortest time possible. Don't hit the walls.

Let's look at the listing.

Line	**Function**
1	Holds machine code for JOYSTICK.
10	Calls JOYSTICK.
20-40	Check for cheese and wall.
50	Erases old mouse, prints new one.
60	Increases time score.
70	Ends main loop.
800	Subroutine—hits wall.
1000	Subroutine—finds cheese.
5000	Begins "Create maze" subroutine.
5020	Sets level of difficulty of maze (closer to 0.0 is harder).
6010	Sets time 0.
6020 and 6030	Set initial PRINT positions to 1,1.
6040	Prints the mouse.

For some people, Mousemaze as listed runs too fast. You can add a delay loop inside the main loop, such as:

```
65 FOR I=1 TO 12
66 NEXT I
```

Decrease the duration of the delay loop as your skill improves and eventually delete these lines altogether.

After you get the cheese, press any key to start the same maze over. To generate a new maze, just press BREAK and then RUN again.

Inverse video appears in these lines:

Line	**Reverse Video**
800	space HITS WALL space
830	graphic shift A
840	asterisk, space, asterisk, space, asterisk, space, asterisk, space
1000	space FINDS CHEESE space
5000	space MAZE space
5060	space
5070	graphic shift A
5100	7 spaces M space O space U space S space E space M space A space Z space E 8 spaces
5140	32 spaces
5200	graphic shift A
5210	two graphic shift A
5220	three graphic shift A
5230	four graphic shift A
5240	five graphic shift A
5250	five graphic shift A
6000	space INITIALIZE space
6080	three graphic shift A

Use the POWER within your reach!

timex sinclair user

A new and unique magazine for the world's most popular personal computer

12 issues only$29.95 Save $5.45! Plus receive free software cassette!*

The best add-on value for your ZX81 or TIMEX SINCLAIR computer

Now that you've got a TIMEX SINCLAIR computer make sure you get the new magazine that helps you to get the most out of it . . . TIMEX SINCLAIR USER! And take advantage of our special bonus of a free software cassette ($15.00 retail value) with each subscription.

TIMEX SINCLAIR USER is published monthly. It keeps you totally up-to-date. It helps you make full use of the power of your computer.

TS1000
ZX81

TIMEX SINCLAIR 1000: the price and technology breakthrough that finally made computers affordable.

TIMEX SINCLAIR USER: the magazine that helps you use the power of this revolutionary computer.

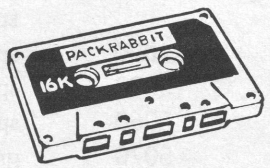
*U.S. dollars

TIMEX SINCLAIR USER covers the full spectrum of user interests. Education. Business applications. Home management. Games and family entertainment. And TIMEX SINCLAIR USER does it in an easy-to-understand style that's authoritative yet friendly.

Each issue is packed with articles and information that help owners make use of the power of their computer. News about software releases. Reviews. Hardware developments. Reader question and answer column. Eight pages of program listings in every issue. New and unique user applications. How to program. Interviews. Special money saving offers to readers.

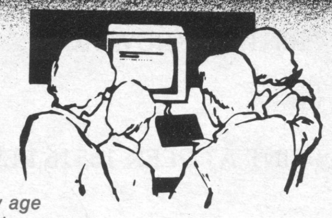
In fact, all the latest ideas and information to ensure that new owners and expert users never tire of their computer.

FREE BONUS WITH YOUR SUBSCRIPTION



"Packrabbit" Retail value $15.00

A good reason to order now! Receive a software program cassette of an exciting, new arcade game! "Packrabbit" is a regular $15.00 retail value. Free with subscriptions for a limited time only.



For any age or experience

Remember: the TS1000 was designed precisely for you. Whatever your age and experience in computers, TIMEX SINCLAIR USER can help you use the power of your computer.

SUBSCRIBE NOW and take advantage of our special offer. Only*$29.95 for 12 monthly issues postage paid. This represents a $6.00 saving over regular newstand prices. And remember, you receive a FREE BONUS with every subscription, a "Packrabbit" software cassette ($15.00 retail value).

To order call

800-543-3000

Ask for Operator 243. In Canada call (513) 729-4300 and ask for Operator 243.

These numbers are for orders only. Have your VISA, Mastercard ready.

Mail to TIMEX SINCLAIR USER, 49 La Salle Avenue, Buffalo, New York, 14217

Name ____________

Address ____________

City ______ State ______ Zip ______

( ) Check ( ) Money order. Or charge to

( ) Visa ( ) Mastercard

Number ____________

Expiry ____________

Signature ____________

timex sinclair user

( ) Yes, I want to subscribe to TIMEX SINCLAIR USER at the special rate of*$29.95 for 12 issues delivered to my door. In addition I understand I will receive the FREE software cassette with my first issue.

AD Code
SQ1

```
   1 REM ▀▀▀ ▀U▙RNDM█RNDU▄RNDM█RN
DLN ▩▀? RETURN COPY COS ACS 74"U
▙RND RETURN +C(UM▙RNDACS 74"U▙RN
D RETURN  C▪XM▙RNDACS 74"U▄RND R
ETURN  C▪XM▄RNDACS ?""U▄RND RETU
RN 3COS UM▄RNDTAN ..
   2 REM MOUSEMAZE
   3 GOTO 5000
  10 RAND USR 16518
  20 PRINT AT PEEK 16514,PEEK 16
515;
  30 LET P=PEEK (PEEK 16398+256*
PEEK 16399)
  35 IF P=40 THEN GOTO 1000
  40 IF P=128 THEN GOTO 800
  50 PRINT AT PEEK 16516,PEEK 16
517;"▒";AT PEEK 16514,PEEK 16515
;"*"
  60 LET S=S+1
  70 GOTO 10
 800 REM [illegible]
 810 LET R=PEEK 16514
 820 LET C=PEEK 16515
 830 PRINT AT PEEK 16516,PEEK 16
517;"▒"
 840 PRINT AT R,C;"▩";AT R,C;"█"
;AT R,C;"▩";AT R,C;"█";AT R,C;"▩
";AT R,C;"█";AT R,C;"▩";AT R,C;
█"
 850 GOTO 6020
1000 REM [illegible]
1005 PRINT AT PEEK 16516,PEEK 16
517;"▒"
1010 FOR I=1 TO 20
1020 PRINT AT 20,30;"▩"
1030 PRINT AT 20,30;"C"
1040 NEXT I
1050 PRINT AT 20,1;S
1060 PRINT AT 20,30;"▒"
1090 GOTO 6000
5000 REM [illegible]
5010 FAST
5020 LET D=.75
5030 RAND
5040 FOR I=1 TO 704
5050 LET Z=RND
5060 IF Z>D THEN PRINT "█";
5070 IF Z<=D THEN PRINT "▒";
5080 NEXT I
5100 PRINT AT 0,0;"[illegible]
[illegible]"
5110 FOR I=1 TO 20
5120 PRINT "█";TAB 31;"█"
5130 NEXT I
5140 PRINT "[illegible]
[illegible]"
5200 PRINT AT 20,29;"▒C"
5210 PRINT AT 19,29;"▒▒"
5220 PRINT AT 1,1;"▒▒▒"
5230 PRINT AT 2,1;"▒▒▒▒▒"
5240 PRINT AT 3,1;"▒▒▒▒▒"
5250 PRINT AT 4,1;"▒▒▒▒▒"
5300 SLOW
6000 REM [illegible]
6010 LET S=0
6020 POKE 16514,1
6030 POKE 16515,1
6040 PRINT AT PEEK 16514,PEEK 16
515;"*"
6060 IF INKEY$="" THEN GOTO 6040
6070 PRINT AT 20,30;"C"
6080 PRINT AT 20,1;"▒▒▒▒"
6100 GOTO 10
9020 NEXT I
SYNTACTIC SUM: 31683, 8K ROM
```

JOYSTICK ASSEMBLY LISTING

For those of you who enjoy machine language programming, here is the assembly language listing of JOYSTICK. You can, of course, modify it to suit your special purposes. For example, JOYSTICK can use PLOT rather than PRINT. Please let me know about your improvements.

Dan Tandberg
4130 Coe Drive NE
Albuquerque, NM 87110

SQ

Address	Hex Code	Label	Assembly
4082	0A	ROW	DATA
4083	0F	COL	DATA
4084	0A	XROW	DATA
4085	0F	XCOL	DATA
4086	3A8240	BEGN	LD A,(ROW)
4089	328440		LD (XROW),A
408C	3A8340		LD A,(COL)
408F	328540		LD (COL),A
4092	CDBB02	NOKEY	CALL 02BB
4095	7D		LD A,L
4096	FEFF		CP FF
4098	C8		RET Z
4099	CB4D	DOWN	BIT 1,L
409B	200B		JR NZ UP
409D	3A8240		LD A,(ROW)
40A0	FE15		CP 15
40A2	2813		JR Z LEFT
40A4	3C		INC A
40A5	328240		LD (ROW),A
40A8	CB55	UP	BIT 2,L
40AA	200B		JR NZ LEFT
40AC	3A8240		LD A,(ROW)
40AF	FE00		CP 00
40B1	2804		JR Z LEFT
40B3	3D		DEC A
40B4	328240		LD (ROW),A
40B7	CB6D	LEFT	BIT 5,L
40B9	200B		JR NZ RIGHT
40BB	3A8340		LD A,(COL)
40BE	FE00		CP 00
40C0	2804		JR Z RIGHT
40C2	3D		DEC A
40C3	328340		LD (COL),A
40C6	CB65	RIGHT	BIT 4,L
40C8	C0		RET NZ
40C9	3A8340		LD A,(COL)
40CC	FE1F		CP 1F
40CE	C8	RET Z	
40CF	3C		INC A
40D0	328340		LD (COL),A
40D3	C9		RET

by David Lipman, Spring Valley, NY

Bond Yield to Maturity

With money market earnings down, some people look to the bond market to earn higher interest rates. New York Stock Exchange (NYSE) Bond listings in newspapers usually report "Current Yield" calculated by dividing the annual coupon income by the bond's purchase price.

Total return (or Yield to Maturity) of a bond bought above or below par (100 = $1000) is impossible to calculate directly. The calculation combines the Present Value (PV) of the annual coupons with the mature bond's PV.

This program assumes (incorrectly) that all bonds listed on the NYSE mature on January 1 and that today's date is January 1.

BOND YIELD TO MATURITY (BYM) guesses at the YTM. Based on the first guess, it converts the coupon cash flow into that flow's PV and adds it to the PV of the mature bond. The program then compares the estimated total value with the actual cost of the bond, adjusts the YTM proportionately and repeats the calculation. With the new PV and interest rate, an approximate tangent to the graphical relation of interest rate and PV can be drawn. For normal values, the tangent intersects the line representing actual cost closer to the actual interest rate.

BYM tests whether values found are within acceptable limits. If not acceptable, it makes a proportionate adjustment in the interest rate, calculates a new PV and a new tangent is drawn. The program repeats the process until it finds an acceptable answer.

This method of solution adapts Sir Isaac Newton's method of using the first derivative of an equation to adjust the iterations. For normal values the program series converges fairly rapidly (5 to 7 steps).

To entertain yourself during solution, run it in SLOW. It prints the initial interest assumption with the corresponding PV and all subsequent interest/PV pairs. Note that alternate adjustments are small resulting from proportionate adjustments and large as a result of drawing a tangent to the curve.

Another reason for running in SLOW is that for extremely large or small rates of return (unlikely in the real world) the program converges slowly enough to make you think your computer crashed.

Using BYM—An Example

To use BYM, input information from the newspaper's bond listings. The first number after the bond name is the annual coupon (rate) and the second is the abbreviated maturity year. Column headings identify the rest of the numbers. Yesterday's closing price might be today's opening price and we'll use that in our examples. See the Wall Street Journal excerpt next page.

Note that AT&T has three bonds maturing near 1990. We'll enter the first, ATT3-7/8s90, converting fractions to decimals. Type in the responses to prompts listed under INPUT. Your computer's answers appear under PRINT:

INPUT	PRINT
ANNUAL COUPON 3.875	$38.75
MATURITY YEAR 90	1990
COST 70.25	$702.50 + ACCRUED INTEREST

```
9.98 PERCENT YIELD TO MATURITY
```

At first glance at the newspaper listing, the three bonds appear to have widely different rates of return, ranging from 5.5 percent to 12 percent. The total yield or yield to maturity range is closer, being:

BOND	YTM
3 7/8 90r	9.98 percent
10 3/8 90	10.48 percent
13 1/4 91	11.71 percent

This information simplifies your investment decisions since you can now deal with the other factors involved in bond choice (current income, future income, or income tax considerations) from a more informed viewpoint.

36 THE WALL STREET JOURNAL, Monday, January 10, 1983

New York Exchange Bonds

Friday, January 7, 1983

Total Volume $38,410,000

SALES SINCE JANUARY 1

1983	1982	1981
$178,809,000	$106,033,000	$97,987,000

	Domestic Fri.	Domestic Thu.	All Issues Fri.	All Issues Thu.
Issues traded	1,067	1,001	1,077	1,013
Advances	604	530	610	537
Declines	253	277	255	280
Unchanged	210	194	212	196
New highs	159	136	163	140
New lows	2	1	2	1

CORPORATION BONDS
Volume, $38,030,000

Bonds	Cur Yld	Vol	High	Low	Close	Net Chg.
ATT 3¼s84	3.5	112	92½	91½	92	+ ¼
ATT 4⅜s85	4.9	63	90⅜	89¾	89¾	− ⅛
ATT 2⅝s86	3.2	37	82	81⅝	81⅝	− ⅜
ATT 2⅞s87	3.7	14	78⅛	78⅛	78⅛	− ½
ATT 3⅞s90	5.5	156	70⅞	70½	70⅞	+ ⅛
ATT 3⅞s90r	5.5	20	70¼	70¼	70¼	
ATT 8¾s00	11.	677	81⅝	81	81	+ ⅛
ATT 7s01	10.	111	68⅞	68¼	68⅝	+ ¼
ATT 7⅛s03	10.	99	68⅞	68¼	68½	+ ¼
ATT 8.80s05	11.	234	80½	80	80	
ATT 8⅝s07	11.	226	78½	78	78¼	+ ¼
ATT 10⅜s90	10.	217	99½	98⅞	99½	+ ¾
ATT 13¼s91	12.	54	107¾	107½	107¾	+ ¼
Ancp 9½s00	cv	65	113	112½	113	+ 1
Ancp 13⅞s02	cv	31	120	118½	120	+ 1⅞

Bonds

NwnBl 9½1
Norton 9½0
Oakln 11⅞s6
Oakln 10½(
OhBIT 7½1
OhEd 9½s06
OhEd 8½s06
OhEd 8⅜s07
OhEd 9½s08
OhEd 15¼8
OhEd 11⅞1
OhEl 11s83
OklGE 4¼9
Orlon 6s88
Orlon 11s98
Orlon 10s99
PSA 11⅛s04
PGE 5s89

Reprinted by permission of Dow Jones & Company, Inc. All rights reserved.

BYM is an abbreviated version of a program that uses purchase, maturity and call dates, coupon rate and purchase price as input, reads out yield to maturity or call and accrued interest payable to seller. This version lacks input check traps. Tapes of both programs are available at $9.95.

David Lipman
9 West Castle Avenue
Spring Valley, NY 10977

Program Summary:

Note that you must change lines 100, 110 and 260 every year on January 1 to match current year.

Line	
10-20	Instructions.
30-60	Input coupon rate, convert to $ per bond, print.
70-190	Input abbreviated year, add century and print.
200-240	Input cost, multiply by 10 to equal cost of one $1000 bond.
250	Set cycle counter to zero.
260	Calculate years to maturity: N.
300-310	Add mature bond value plus all coupons, compare with cost to determine if there is a percent profit.
400	Calculate initial Io percent return.
420	Subr. - calculate PV=X to match Io.
500-510	Store Io, Xo for future use.
600	Adjust Io by percent difference between Xo and cost.
620	Subr. - calculate new PV to match new I.
700-710	Draw tangent to curve to determine new I.
720	Subr. - calculate new PV to match new I.
800	Loop to 500.
900-1200	Print instructions for solutions.
1300	Stop.
2000-2010	Subroutine to calculate present value of mature bond plus annual coupons.
2020	Print instructions to show intermediate calculations so you don't get bored waiting.
2100-2120	Precision limits 1/2 cent or 1/100 percent.
2200	Cycle counter.
2210	If cycle exceeds 9, scroll to leave room for print.
2500	Subroutine end, return to main program.

RUNNING TIME: Slow: 10-20 seconds

Fast: 1-3 seconds if you eliminate intermediate print steps by LIST ENTER, 2020 ENTER.

LIMITATIONS: Will not run if current return is under 0.01 percent. Solve zero coupon bonds by formula

$$I=\left[\left(\frac{\text{Mature Value}}{\text{Cost}}\right)^{1/n}-1\right]100$$

EXIT: During data input use STOP, ENTER. While running use BREAK.

FORMULA USED:

$$COST=1000(1+i)^{-n}+\text{Coupon}\ \frac{(1+i)^{n}-1}{i\,(1+i)^{n}}$$

Solve for i. See Grant's "Principles of Engineering Economy" 1938.

If "out of memory" report (code 5) appears at lower left of screen, press CONT to go on. **SQ**

```
  10 PRINT "PROGRAM TO ESTIMATE
YIELD TO  MATURITY OF BOND BOU
GHT ABOVE OR BELOW PAR"
  20 PRINT " INPUT DATA AS IN NE
WSPAPER"
  30 PRINT
  40 PRINT " INPUT  ANNUAL COUPO
N  ";
  50 INPUT C
  60 PRINT "$";C*10
  70 PRINT
  80 PRINT " INPUT MATURITY YEAR
";
  90 INPUT B
 100 IF B<83 THEN GOTO 150
 110 IF B>83 THEN PRINT "19";B
 120 LET A=B
 130 PRINT
 140 GOTO 205
 150 IF B>9 THEN GOTO 180
 160 PRINT "200";B
 170 GOTO 190
 180 PRINT "20";B
 190 LET A=100+B
 200 PRINT
 210 PRINT " INPUT COST  ";
 220 INPUT PV
 230 PRINT "$";PV*10;" + ACCRUED
 INT."
 240 PRINT
 250 LET S=0
 260 LET N=A-83
 300 LET Q=PV-(N*C)-100
 310 IF Q>=0 THEN GOTO 1100
 400 LET I=C/PV
 420 GOSUB 2000
 500 LET M=I
 510 LET J=X
 600 LET I=(X/PV)*I
 620 GOSUB 2000
 700 LET L=M-(M-I)*(PV-J)/(X-J)
 710 LET I=L
 720 GOSUB 2000
 800 GOTO 500
 900 SCROLL
 910 PRINT
 920 SCROLL
1000 PRINT ABS I*100 ; " PERCENT
  YIELD  TO "
1010 SCROLL
1020 PRINT "MATURITY"
1030 GOTO 1160
1100 PRINT
1110 PRINT "PURCHASE  PRICE  EXC
EEDS  TOTAL COUPONS  PLUS  MATUR
E  VALUE OF BOND RESULTING IN A
NET LOSS OF ";Q*10;" DOLLARS"
1160 SCROLL
1180 PRINT
1190 SCROLL
1200 PRINT " TO CONT PRESS RUN A
ND ENTER"
1300 STOP
2000 LET Y=1/(1+I)**N
2010 LET X=(100*Y)+(C*((1-Y)/I))
2020 PRINT I*100,X*10
2100 LET Z=PV-X
2110 IF ABS Z<.0005 THEN GOTO 90
0
2120 IF ABS I<.0001 THEN GOTO 90
0
2200 LET S=S+1
2210 IF S>10 THEN SCROLL
2500 RETURN
SYNTACTIC SUM:  3067, 8K ROM
```


"Far and away the best book for ZX81 [TS 1000] users new to computing."
—Popular Computing Weekly

TIMEX SINCLAIR 1000
Programs, Games, and Graphics
by Ian Stewart and Robin Jones
156 pp. Softcover

...and from the same authors...

NEW

Machine Code and Better BASIC
(For the TS 1000 and TS 2000)
by Ian Stewart and Robin Jones
188 pp. Softcover

"...beautifully written...joyful, readable, user-friendly..."
—Education Equipment

ORDER NOW AND RECEIVE A 10% DISCOUNT!
(Order **must** be placed on this coupon, and must be received by July 31, 1983 to be valid.)

Send me____copies of **Timex Sinclair 1000** at **$9.85** (list price $10.95).

Send me____copies of **Machine Code and Better BASIC** at **$10.75** (list price $11.95).

Find my payment of________ enclosed.

Payment is by ☐ check ☐ VISA ☐ Mastercard

card no.:________________ expiration date:________________

signature:________________
Please print:
Name:________________
Address:________________
City/State/Zip:________________
Detach and mail to: Birkhauser Boston, P.O. Box 2007, Cambridge, MA 02139
Mass. residents please add 5% sales tax. Postage and handling will be paid by publisher.

by James Eischen, Windsor, CT

8K ROM/16K RAM

555 Timer Calculations

If you're an electronics buff, you've probably had this problem. When I try to calculate resistors values for a 555 timer, I think, "There must be a way around all this frustrating math." After all, calculating is what a computer does best. With this program, 555 TIMER, these calculations are a snap.

After typing in 555 TIMER, enter GOTO 9600 to get the Syntactic Sum (lines 9600-9604 are Lloyd Painter's BASIC Syntactic Sum, SYNTAX, Feb. 83). You should get 305058 using this version and 50237 using SYNTAX's machine code version. Then enter RUN. When the main menu appears, choose option three to SAVE 555 TIMER on tape. The screen clears and you see:
IF YOU WANT TO SAVE, "555 TIMER" - PRESS "1"
IF YOU WANT THE MENU - PRESS "2"
These two lines safeguard 555 TIMER from accidental input. Start your recorder and press 1. 555 TIMER saves to tape. Make several copies. Now whenever you load from tape, 555 TIMER comes up running.

After it finishes saving, the program returns you to the main menu. Just press the number of the program you want. When you select the astable mode, the computer requests the capacitor size you want to use, the frequency and the duty cycle as a percent. Don't enter duty cycles less than 50 percent or 555 TIMER will compute negative resistor values. Remember that duty cycle is the time output must be low compared to total time. The program converts the duty cycle to a decimal. The display shows the astable drawing with three examples. If the resistors are not near a standard size, press Y to recalculate, then go through the program again with another size capacitor until the resistors are at or near standard size.

When you select the monostable mode, the computer requests the capacitor size and time delay. Enter the time as seconds, for example, 50 ms = .05 seconds. The display shows the monostable drawing with two examples. Again, as in the astable mode, select Y to recalculate to obtain a standard size resistor. Select N to get back to the main menu.

Variables List:

C = Capacitor
F = Frequency
K = C/1000
D = Duty Cycle (Astable mode)
T = Time Delay (Monostable mode)

555 TIMER contains CLEAR statements so that the variables will not conflict with either program selected.

Inverse graphics appear in the following lines:

Line

6 K$ = " " graphics shift space 32 times
8 555 TIMER
11 graphics shift space, graphics shift space, graphics shift 5, graphics shift 8
316 555 TIMER-ASTABLE MODE
396 ASTABLE
672 MONOSTABLE
712 555 TIMER-MONOSTABLE MODE

SQ

```
   4 CLEAR
   5 FAST
   6 LET K$="████████████████████████████████"
   7 PRINT AT 0,0;K$;AT 1,0;K$;AT 2,0;K$
   8 PRINT AT 1,11;"555 TIMER"
   9 FOR N=1 TO 9
  11 PRINT "█";TAB 31;"█";TAB 0;"▌";TAB 31;"▐"
  12 NEXT N
  15 PRINT K$;TAB 0;K$
  16 PRINT AT 7,7;"1. ASTABLE MODE"
  17 PRINT AT 11,7;"2. MONOSTABLE MODE"
  18 PRINT AT 15,7;"3. SAVE ""555 TIMER"""
  26 SLOW
  30 IF INKEY$<>"" THEN GOTO 30
```

Continued Next Page

```
  31 IF INKEY$="" THEN GOTO 31
  32 LET A$=INKEY$
  33 IF A$="1" OR A$="2" OR A$="3" THEN GOTO 35
  34 GOTO 30
  35 LET Q=VAL A$
  36 GOSUB Q*300
  37 CLS
  38 GOTO 4
 300 GOSUB 314
 301 PRINT ,,"TYPE CAPACITOR SIZE (UF)"
 302 INPUT C
 303 IF C<=.01 THEN GOTO 301
 305 GOSUB 314
 307 PRINT ,,"TYPE FREQUENCE (HZ)"
 308 INPUT F
 309 GOSUB 314
 311 PRINT ,,"TYPE DUTY CYCLE (PERCENT)"
 312 INPUT D
 313 GOTO 319
 314 FAST
 315 CLS
 316 PRINT "555 TIMER-ASTABLE MODE"
 317 SLOW
 318 RETURN
 319 CLS
 320 LET P=D/100
 321 LET K=C/1000
 322 LET RA=(2*P-1)/(.693*K*F)
 323 LET RB=((1/(1.386*K*F))-(RA/2))
 324 LET RA=INT (RA*100+.5)/100
 325 LET RB=INT (RB*100+.5)/100
 329 FAST
 330 FOR N=9 TO 17
 331 PLOT N,29
 332 NEXT N
 333 FOR N=9 TO 17
 334 PLOT N,16
 335 NEXT N
 336 FOR N=16 TO 29
 337 PLOT 17,N
 338 NEXT N
 339 FOR N=16 TO 29
 340 PLOT 9,N
 341 NEXT N
 342 FOR N=18 TO 25
 343 PLOT N,27
 344 NEXT N
 345 FOR N=25 TO 29
 346 PLOT 26,N
 347 NEXT N
 348 FOR N=30 TO 33
 349 PLOT 25,N
 350 NEXT N
 351 FOR N=30 TO 33
 352 PLOT 27,N
 353 NEXT N
 354 FOR N=34 TO 37
 355 PLOT 26,N
 356 NEXT N
 357 FOR N=21 TO 24
 358 PLOT 25,N
 359 NEXT N
 360 FOR N=21 TO 24
 361 PLOT 27,N
 362 NEXT N
 363 FOR N=15 TO 20
 364 PLOT 26,N
 365 NEXT N
 366 FOR N=18 TO 25
 367 PLOT N,18
 368 NEXT N
 369 FOR N=25 TO 27
 370 PLOT N,14
 371 NEXT N
 372 FOR N=25 TO 27
 373 PLOT N,12
 374 NEXT N
 375 FOR N=9 TO 11
 376 PLOT 26,N
 377 NEXT N
 378 FOR N=24 TO 28
 379 PLOT N,8
 380 NEXT N
 381 FOR N=25 TO 27
 382 PLOT N,6
 383 NEXT N
 384 FOR N=3 TO 8
 385 PLOT N,25
 386 NEXT N
 387 FOR N=15 TO 19
 388 PLOT N,13
 389 NEXT N
 390 FOR N=13 TO 15
 391 PLOT 15,N
 392 NEXT N
 393 FOR N=13 TO 17
 394 PLOT 20,N
 395 NEXT N
 396 PRINT AT 0,0;"ASTABLE"
 397 PRINT AT 2,13;"+VCC";AT 5,15;"RA";AT 7,9;"7";AT 8,3;"3"
 398 PRINT AT 6,15;RA;" K-OHMS"
 399 PRINT AT 9,6;"5";AT 9,15;"RB";AT 10,6;"5";AT 10,15;RB;" K-OHMS"
 400 PRINT AT 11,6;"5";AT 11,9;"6";AT14,6;"2";AT 14,15;"C"
 401 PRINT AT 15,15;C;" UF";AT 17,15;"FREQ.= ";F;" HZ"
 402 PRINT AT 16,0;"DO U WANT TO ";TAB 0;"RECALCULATE";TAB 0;"THIS MODE?";TAB 0;"Y OR N"
 403 PRINT AT 18,15;"DUTY CYCLE= ";P
 408 SLOW
 410 IF INKEY$<>"" THEN GOTO 410
 411 IF INKEY$="" THEN GOTO 411
 412 LET B$=INKEY$
 413 IF B$="Y" THEN GOTO 416
 414 IF B$="N" THEN RETURN
 415 GOTO 410
 416 GOSUB 710
 417 PRINT ,,"FREQ. IS STILL ";F;" HZ"
 418 PRINT ,,"DUTY CYCLE IS STILL ";P
 419 PRINT ,,,,"TYPE CAPACITOR SIZE (UF)"
 420 INPUT C
 421 GOTO 319
 602 GOSUB 710
 603 PRINT ,,"TYPE TIME DELAY IN ","SECONDS(IE;.05 SECONDS=50 MS.)"
 604 INPUT T
 605 GOSUB 710
 607 PRINT ,,"TYPE CAPACITOR SIZE (UF)"
 608 INPUT C
 609 IF C<=.001 THEN GOTO 607
 610 CLS
 612 LET R=INT (((1000*T)/(1.1*C))*100+.5)/100
```

Continued Next Page

```
615 FAST
616 FOR N=9 TO 17
617 PLOT N,29
618 NEXT N
619 FOR N=9 TO 17
620 PLOT N,18
621 NEXT N
622 FOR N=18 TO 28
623 PLOT 9,N
624 NEXT N
625 FOR N=18 TO 28
626 PLOT 17,N
627 NEXT N
628 FOR N=2 TO 9
629 PLOT N,25
630 NEXT N
631 FOR N=18 TO 28
632 PLOT N,27
633 NEXT N
634 FOR N=18 TO 23
635 PLOT N,21
636 NEXT N
637 FOR N=21 TO 26
638 PLOT 24,N
639 NEXT N
640 FOR N=18 TO 30
641 PLOT 29,N
642 NEXT N
643 FOR N=31 TO 36
644 PLOT 28,N
645 NEXT N
646 FOR N=31 TO 36
647 PLOT 30,N
648 NEXT N
649 FOR N=37 TO 41
650 PLOT 29,N
651 NEXT N
652 FOR N=28 TO 30
653 PLOT N,17
654 NEXT N
655 FOR N=28 TO 30
656 PLOT N,15
657 NEXT N
658 FOR N=9 TO 14
659 PLOT 29,N
660 NEXT N
661 FOR N=26 TO 32
662 PLOT N,8
663 NEXT N
664 FOR N=27 TO 31
665 PLOT N,6
666 NEXT N
667 FOR N=28 TO 30
668 PLOT N,4
669 NEXT N
672 PRINT AT 0,0;"MONOSTABLE";A
T 0,14;"+VCC";AT 4,16;"R";TAB 16
;R;" K-OHMS"
673 PRINT AT 7,9;"7";AT 7,16;"T
IME DELAY= ";T;AT 8,3;"3";AT 8,6
;"5";AT 8,16;" SECONDS"
674 PRINT AT 9,6;"5";TAB 6;"5";
AT 10,9;"6"
675 PRINT AT 13,16;"C";TAB 16;C
;" UF"
677 PRINT AT 7,16;"TIME DELAY=
";T;TAB 16; "SECONDS."
689 PRINT AT 13,0;"DO U WANT TO
";TAB 0;"RECALCULATE";TAB 0;"THI
S MODE?";TAB 0;"Y OR N"
691 SLOW
693 IF INKEY$<>"" THEN GOTO 693
694 IF INKEY$="" THEN GOTO 694
695 LET C$=INKEY$
696 IF C$="Y" THEN GOTO 720
697 IF C$="N" THEN RETURN
698 GOTO 693
710 CLS
711 FAST
712 PRINT "555 TIME-MONOSTABLE
MODE"
713 SLOW
714 RETURN
720 GOSUB 710
721 PRINT ,,"TIME DELAY IS STIL
L ";T;" SECONDS"
722 GOTO 607
899 STOP
900 CLEAR
903 CLS
904 PRINT ,,"IF YOU WANT TO SAV
E, 555 TIMER - PRESS ""1"""
905 PRINT ,,"IF YOU WANT THE ME
NU - PRESS ""2"""
908 IF INKEY$<>"" THEN GOTO 908
909 IF INKEY$="" THEN GOTO 909
910 LET S$=INKEY$
911 IF S$="1" THEN GOTO 914
912 IF S$="2" THEN RETURN
913 GOTO 908
914 CLS
915 CLEAR
916 SAVE "555 TIMER"
917 GOTO 5
9600 LET S=0
9601 FOR A=16509 TO PEEK 16396+2
56*PEEK 16397-1
9602 LET S=S+PEEK A
9603 NEXT A
9604 PRINT "S=";S-7323
SYNTACTIC SUM: 50237 8K ROM
```


Available from SYNTAX...

For computing beginners —

Crash Course in Microcomputers **$19.95**
Covers hardware, machine language and applications. Reviewed in *SYNTAX*, Oct. 1981. Add $1.50 shipping.

ZX80 Pocket Book **$10.95**
Includes ZX81 supplement. Covers Sinclair BASIC, data and program listings. Add $1.50 postage.

For advanced hardware/software users —

Zilog's Z80-Z80A CPU Technical Manual **$7.88**

Zilog's Assembly Language Programming **$15.75**

Experiments in Artificial Intelligence **$9.95**
Add $1.50 postage.

SYNTAX back issues available, $4 each.
Call or write for our group subscription discounts.

SYNTAX • RD 2 Box 457 • Harvard, MA 01451
617 / 456-3661

by Herman Geschwind, Greensboro, NC

8K ROM/16K RAM

Federal Income Tax Planner

In this time of "bracket creep," rapidly changing interest rates, significant changes in tax rates, and limitations on previously generally accepted deductible items, tax planning is more important to all of us than ever. While taxes, like death, are inevitable, a well founded tax strategy may substantially reduce or defer your current and future taxes.

Using your ZX/TS's computational capability makes playing through alternative strategies and "what if" calculations not a chore but a pleasure. It's even easier if you already use your computer to track your monthly financial status. See "Household Finance" by James W. Holder, SYNTAX, Jul. 82, p.4. It's easy to annualize the output of Holder's program and to pass the data to TAX PLANNER to test the impact of the Federal Income Tax on your personal finances.

TAX PLANNER lets you quickly and easily evaluate such questions as:

- What is the real impact of the scheduled tax reduction on my financial situation?
- How does a change in filing status affect me?
- What happens if personal exemptions change?
- With declining interest rates, is it still a good strategy to borrow against equity (incurring an interest expense deduction) to take advantage of the borrowed money invested elsewhere?
- How much is an IRA account worth to me now?

TAX PLANNER automatically recognizes:

- Change in treatment of medical insurance premium in 1983.
- Change in limitation of medical expense deduction in 1983.
- Appropriate limitations on deductions ("Zero Bracket") depending on the filing status you enter.
- The 25 percent limit on charitable deductions if you do not otherwise itemize deductions.

TAX PLANNER also automatically loads appropriate tax rate schedules depending on filing status entered.

TAX PLANNER displays results on your screen and also directs appropriate output to an attached printer.

```
   1 REM TAX PLANNER
   2 REM ...COPYRIGHT..1982.....

   3 REM ...BY..H.GESCHWIND.....
 100 REM INITIALIZE
 110 DIM D$(25,18)
 120 DIM S(25)
 130 DIM T(25)
 135 LET M$="PRESS ANY KEY TO CO
NTINUE"
 140 LET P$="ENTER"
 150 LET X$="-------------------
-------------"
 160 LET Y$=" 1982"
 170 LET Z$=" 1983"
 180 LET V$="INCOME"
 190 LET W$="DEDUCTIONS"
 195 LET H$="---->"
 200 LET S=0
 210 LET M=0
 220 LET N=0
 225 LET F$=" ENTER FILING STATU
S:-->1 FOR SINGLE,-->2 FOR MARRI
ED,-->3 FOR MARRIED FILING SEPAR
ATELY"
 240 GOSUB 1000
 242 LIST 999
 499 REM MENU
 500 PRINT AT 3,6;"    TAX PLAN
NER"
 510 PRINT AT 5,4;"-->PRESS DESI
RED FUNCTION"
 520 PRINT AT 8,6;"1. ENTER VALU
ES"
 525 PRINT AT 9,6;"2. REVIEW RES
ULTS"
 530 PRINT AT 10,6;"3. CLEAR DAT
A"
 570 PRINT AT 11,6;"4. SAVE DATA
"

 580 PRINT AT 12,6;"5. ENTER 198
3 VALUES ONLY"
 590 PAUSE 4E4
 600 LET K$=INKEY$
 610 IF K$<"1" OR K$>"5" THEN GO
TO 690
```

Continued Next Page

Printed output is especially desirable if you want to compare the results of alternative assumptions. However, while TAX PLANNER uses official IRS Tax Rate Schedules for computing tax, it doesn't calculate the actual tax liability for filing your return! To file your return use "Income Tax" by Lane Lester, SYNTAX, Mar. 82, page 8.

To keep within the limitations of the ZX/TS 16K memory, TAX PLANNER does not provide for the calculation of Alternative Minimum Tax. If AMT might be applicable to you, consult appropriate literature or tax counsel.

You can input salaries and wages up to $9,000,000 and miscellaneous income and deductions up to $99,999.

Program Entry Hints

While TAX PLANNER's program listing appears formidable, you can save much effort in entering the program lines using the ZX/TS EDIT function.

First key in the main body of the program and all program lines pertaining to 1982 (lines 1-3280, 3999-5500, 5999-6900 and 9500-9560).

Create the subroutine 83 VALUES (lines 3499-3998) by editing the 82 VALUES subroutine (lines 2999-3280). Where necessary substitute string variable Z$ for Y$, and array element T for S.

You can also make the subroutine 83 TABLE (lines 5510-5771) from the 82 subroutine (lines 4999-5361). Substitute arrays D, E and F for arrays A, B and C.

Again, the tax calculation routine in lines 5399-5500 is identical to the routine in lines 5799-5900 except for array and variable references.

You can enter the values needed to load the tables (lines 7000-7999) expeditiously by keying in the first complete line, then LIST-EDITing in this line, incrementing the line number and DELETEing and keying in array subscripts and changed values. Proofread the tax values thoroughly to avoid errors in tax calculation!

Installation

After keying the listing, press RUN and ENTER to initialize the program. The program loads descriptions of all line items into an array. This array subsequently provides input prompts and output listings. After an initial RUN the program stops and the descriptive loader portion of the program listing (starting with line 999) appears on the screen. Now DELETE lines 999-1998. (If you forget this step, you'll get an "Out of Memory" error 5 later on!)

Press GOTO 500 to return to the Main Menu. Select option 4 to save the run-time version of TAX PLANNER, the version you use from now on.

All subsequent processing automatically returns you to the Main Menu. Using option 4 automatically saves program and variables. **FROM NOW ON DO NOT USE "RUN" TO START PROGRAM!** (Because of TAX PLANNER's length, you need at least a C-20 cassette tape to save it.)

Lines 6054 and 6300 direct output to a CAI/P40 printer. For a printer using keyboard printer commands, like Sinclair's ZX printer, EDIT both lines to read COPY. If you lack a printer, delete lines 6054 and 6300.

```
 620  IF K$="1" THEN GOTO 3000
 622  IF K$="2" THEN GOTO 6000
 624  IF K$="3" THEN GOTO 2000
 626  IF K$="4" THEN GOTO 4000
 627  IF K$="5" THEN GOTO 3500
 690  GOTO 500
 998  STOP
 999  REM DESCRIPTIONS
1000  CLS
1005  PRINT AT 21,0;"LOADING DESC
RIPTIONS"
1010  FOR I=1 TO 26
1020  LET D$(1)="SALARIES, WAGES"
1021  LET D$(2)="DIVIDENDS"
1022  LET D$(3)="INTEREST INCOME"
1023  LET D$(4)="NET BUS. INCOME"
1024  LET D$(5)="CAPITAL GAINS"
1025  LET D$(6)="OTHER GAINS/LOSS
ES"
1026  LET D$(7)="PENSIONS,ANNUITI
ES"
1027  LET D$(8)="RENTS, ROYALTIES
"
1028  LET D$(9)="PARTNERSHIP INC.
"
1029  LET D$(10)="TRUST/ESTATE IN
C"
1030  LET D$(11)="CHAPTER""S""COR
P."
1031  LET D$(12)="OTHER INCOME"
1032  LET D$(13)="INCOME CREDITS(
*)"
1033  LET D$(14)="ADJ. GROSS INCO
ME"
1034  LET D$(15)="MEDICAL EXPENSE
S"
1035  LET D$(16)="TAXES"
1036  LET D$(17)="CONTRIBUTIONS"
1037  LET D$(18)="INTEREST EXPENS
E"
1038  LET D$(19)="CASUALTY LOSS,M
ISC."
1039  LET D$(20)="TOTAL DEDUCTION
S"
1040  LET D$(21)="ZERO BRACKET"
1041  LET D$(22)="NET DEDUCTIONS"
1042  LET D$(23)="INC.BEF.EXEMPTI
ONS"
1043  LET D$(24)="EXEMPTIONS"
1044  LET D$(25)="TAXABLE INCOME"
1045  LET D$(26)="INCOME TAX"
1046  NEXT I
1050  CLS
1060  RETURN
1998  STOP
1999  REM DELETE
2000  CLS
2010  PRINT AT 6,4;"-->PRESS DESI
RED FUNCTION"
2020  PRINT AT 8,6;"1. DELETE 198
2 VALUES"
2030  PRINT AT 9,6;"2. DELETE 198
3 VALUES"
2050  PAUSE 4E4
2060  LET K$=INKEY$
2070  IF K$<"1" OR K$>"2" THEN GO
TO 2090
2072  IF K$="1" THEN GOTO 2100
2074  IF K$="2" THEN GOTO 2300
2090  GOTO 2010
2100  DIM S(26)
2101  LET S=0
2105  CLS
```

Continued Next Page

Entering Values

Option 1 of the Main Menu invokes system prompts to enter values for 1982.

Input and ENTER appropriate full-dollar values. The screen echoes your input and pauses for three seconds to let you review your data.

Enter miscellaneous loss items (Partnership Income, Rents, Royalties, etc.) with a negative sign.

Enter all other deduction and income credits (such as IRA and KEOGH deposits) as positive values. The program recognizes their negative (credit) nature.

Income credits—KEOGH, IRA, alimony and "marriage penalty deduction" (5 percent of the earnings of the lower earning spouse for 1982 and 10 percent for 1983 with a $1500 limit for 1982 and a $3000 limit for 1983)— should be lumped together as one entry.

Enter 1982 medical insurance premiums as *full* amount. The program automatically calculates the lesser of either half the premium or $150 maximum for 1982. When TAX PLANNER prompts you for 1982 medical expenses, enter the total of your medical bills, full insurance premiums and any amount spent for drugs exceeding 1 percent of your adjusted gross income. Note that you enter your medical insurance premiums twice.

After entering 1982 values, the system loads the appropriate tax schedule and then continues prompts for 1983 values. Start by entering your filing status for 1983. Except for medical insurance all other prompts are identical to those for 1982.

After loading 1983 tax rate schedule, the system returns to the Main Menu where you can now call option 2, "Review Results."

To explore alternative strategies, try option 5, "Enter 1983 Values Only." This option lets you retain 1982 values as a base for comparison to alternative values for 1983. To make sure that previously entered values are properly cleared first enter option 3, "Clear Data," with appropriate sub-menu.

Output

Main Menu option 2 displays results on the screen and echoes output on a printer, if attached.

This first display shows your income summary and remains until you press any key, which clears the screen and displays subsequent line items.

Screen two details deductions (adjusted for overall limitations), appropriate "Zero Bracket amounts," net taxable income, and calculated income tax. This screen ends with error report 5.

Press CONT and ENTER, then any other key to return to Main Menu.

At this point, you can SAVE results or CLEAR previous entries and restart by entering alternative assumptions.

Maintenance

While TAX PLANNER is based upon 1982 and 1983 tax rate schedules and tax provisions, you can easily adapt it for use in later years. Program entry points for maintenance are as follows:

Lines 160 and 170 set up string variables for system prompts and output. Change literals.

```
2110 GOTO 500
2300 DIM T(26)
2301 LET S=0
2305 CLS
2310 GOTO 500
2999 REM [illegible]
3000 CLS
3001 LET M=0
3002 LET N=0
3003 LET I=0
3005 PRINT F$
3006 INPUT S
3007 IF S<1 OR S>3 THEN GOTO 300
5
3008 CLS
3010 FOR I=1 TO 13
3015 PRINT AT 10,(31-LEN V$)/2;V
$
3016 PRINT X$
3020 PRINT P$
3025 PRINT D$(I)+" FOR "+Y$
3030 INPUT S(I)
3035 IF S(13) THEN LET S(13)=-1*
S(13)
3036 PRINT H$; S(I)
3037 PAUSE 180
3038 CLS
3039 LET M=M+S(I)
3040 NEXT I
3041 LET S(14)=M
3042 LET M=0
3050 PRINT AT 10,(31-LEN U$)/2;U
$
3051 PRINT X$
3052 PRINT P$+" FULL AMOUNT OF M
EDICAL        INSURANCE PREMIUMS"
3053 INPUT I
3054 PRINT H$;I
3055 PAUSE 180
3056 CLS
3058 LET I=I/2
3060 IF I>150 THEN LET I=150
3062 PRINT AT 10,(31-LEN W$)/2;W
$
3064 PRINT X$
3066 PRINT P$
3068 PRINT D$(15)+" FOR "+Y$
3102 INPUT N
3103 PRINT H$;N
3104 LET M=INT (3/100*S(14)+.5)
3105 LET S(15)=N-M
3106 IF S(15)<=0 THEN LET S(15)=
0+I
3107 LET M=0
3108 LET N=0
3109 CLS
3110 FOR I=16 TO 19
3111 PRINT AT 10,(31-LEN W$)/2;W
$
3112 PRINT X$
3120 PRINT P$
3125 PRINT D$(I)+" FOR "+Y$
3130 INPUT S(I)
3140 PRINT H$;S(I)
3150 PAUSE 180
3160 CLS
3170 LET M=M+S(I)
3180 NEXT I
3185 IF S(15)=0 AND S(16)=0 AND
S(18)=0 AND S(19)=0 THEN LET S(1
7)=S(17)/4
3190 LET S(20)=M+S(15)
```

Continued Next Page

Lines 3050-3060 capture medical insurance premiums for 1982. Delete if necessary.

Line 3104 calculates overall limitation on medical deductions. Change percentage value as appropriate.

In line 3106 delete variable I (one-half of insurance premium).

In subroutine lines 6900-7900 (Tax Rate Schedules 82) arrays A, B and C are for brackets, tax amounts and percentages for first year. Similarly, arrays D, E and F are for second year. Edit D, E and F lines to be first year and enter new items for second year. Use IRS publications to find appropriate schedules.

*(Long programs such as TAX PLANNER increase your risk of losing data bits while LOADing. We suggest you use "Syntactic Sum with Variables" (**SQ**, Winter 1982) to check TAX PLANNER or any long program for a good load each time you use it. The Syntactic Sum given is for TAX PLANNER exactly as listed, before deleting lines 999-1998.—Ed.)*

To help you enter TAX PLANNER, here's a list of the words in reverse video in this listing:

Line	**Reverse video**
1	TAX PLANNER
100	INITIALIZE
499	MENU
999	DESCRIPTIONS
1999	CLEAR
2999	82 VALUES
3499	83 VALUES
3999	SAVE
4999	82 TABLE
5399	82 CALC
5510	83 TABLE
5799	83 CALC
5999	PRINT RESULTS
6900	82 TAX
7400	83 TAX
9500	JUSTIFY

SQ

```
3192 IF S(15)=0 AND S(16)=0 AND
S(18)=0 AND S(19)=0 THEN LET S(2
0)=S(17)
3195 LET M=0
3200 IF S=1 THEN LET S(21)=2300
3205 IF S=2 THEN LET S(21)=3400
3210 IF S=3 THEN LET S(21)=1700
3215 IF S=3 THEN LET S=1
3220 LET S(22)=S(20)-S(21)
3225 IF S(22)<=0 THEN LET S(22)=
0
3230 LET S(23)=S(14)-S(22)
3240 PRINT "ENTER NUMBER OF EXEM
PTIONS"
3250 INPUT M
3255 IF M<1 THEN LET M=1
3260 LET S(24)=M*1000
3265 LET M=0
3270 LET S(25)=S(23)-S(24)
3275 CLS
3280 GOSUB 5000
3499 REM 83 VALUES
3500 CLS
3501 LET M=0
3502 LET N=0
3505 PRINT F$
3506 INPUT S
3507 IF S<1 OR S>3 THEN GOTO 350
5
3508 CLS
3510 FOR I=1 TO 13
3515 PRINT AT 10,(31-LEN V$)/2;V
$
3516 PRINT X$
3520 PRINT P$
3525 PRINT D$(I)+" FOR "+Z$
3538 INPUT T(I)
3539 IF T(13) THEN LET T(13)=-1*
T(13)
3540 PRINT H$;T(I)
3550 PAUSE 180
3560 CLS
3570 LET N=N+T(I)
3580 NEXT I
3590 LET T(14)=N
3595 LET N=0
3596 PRINT AT 10,(31-LEN W$)/2;W
$
3597 PRINT X$
3598 PRINT P$
3599 PRINT D$(15)+" FOR "+Z$
3602 INPUT N
3603 PRINT H$;N
3604 LET M=INT (5/100*T(14)+.5)
3605 LET T(15)=N-M
3606 IF T(15)<=0 THEN LET T(15)=
0
3607 LET M=0
3608 LET N=0
3609 CLS
3610 FOR I=16 TO 19
3611 PRINT AT 10,(31-LEN W$)/2;W
$
3612 PRINT X$
3620 PRINT P$
3625 PRINT D$(I)+" FOR "+Z$
3630 INPUT T(I)
3640 PRINT H$;T(I)
3650 PAUSE 180
3660 CLS
3670 LET N=N+T(I)
3680 NEXT I
3685 IF T(15)=0 AND T(16)=0 AND
T(18)=0 AND T(19)=0 THEN LET T(1
7)=T(17)/4
3690 LET T(20)=N+T(15)
3692 IF T(15)=0 AND T(16)=0 AND
T(18)=0 AND T(19)=0 THEN LET T(2
0)=T(17)
3695 LET N=0
3700 IF S=1 THEN LET T(21)=2300
3705 IF S=2 THEN LET T(21)=3400
3710 IF S=3 THEN LET T(21)=1700
3715 IF S=3 THEN LET S=1
3717 LET M=0
3718 LET N=0
3720 LET T(22)=T(20)-T(21)
3725 IF T(22)<=0 THEN LET T(22)=
0
3730 LET T(23)=T(14)-T(22)
3740 PRINT "ENTER NUMBER OF EXEM
PTIONS"
3750 INPUT M
3755 IF M<1 THEN LET M=1
3760 LET T(24)=M*1000
3765 LET M=0
3770 LET T(25)=T(23)-T(24)
3780 GOSUB 5500
```

Continued Next Page

```
3785 RETURN
3998 STOP
3999 REM SAVE
4000 CLS
4020 PRINT AT 8,7;"PREPARE RECOR
DER"
4030 PRINT AT 10,3;M$
4040 PAUSE 4E4
4050 CLS
4060 SAVE "TAX PLANNER"
4070 GOTO 500
4999 REM 82 TABLE
5000 CLS
5010 IF S=1 THEN LET B=13
5011 IF S=2 THEN LET B=12
5040 DIM A(B)
5050 DIM B(B)
5060 DIM C(B)
5065 PRINT AT 21,0;"LOADING TAX
TABLE"
5070 FOR I=1 TO B
5075 GOSUB 6900
5180 NEXT I
5350 GOTO 5399
5361 STOP
5399 REM 82 CALC
5400 LET S(26)=0
5420 IF S(25)<A(1) THEN RETURN
5430 FOR E=2 TO B
5440 IF S(25)<A(E) THEN GOTO 546
0
5450 NEXT E
5460 LET E=E-1
5470 LET S(26)=INT ((B(E)+(S(25)
-A(E))*C(E)/100)+.5)
5480 LET B=0
5490 RETURN
5491 STOP
5500 CLS
5510 REM 83 TABLE
5535 LET B=13
5540 DIM D(B)
5550 DIM E(B)
5560 DIM F(B)
5565 PRINT AT 21,0;"LOADING TAX
TABLE"
5570 FOR I=1 TO B
5580 GOSUB 7400
5690 NEXT I
5770 GOTO 5800
5771 STOP
5799 REM 83 CALC
5800 LET T(26)=0
5810 IF T(25)<D(1) THEN RETURN
5820 FOR E=2 TO B
5830 IF T(25)<D(E) THEN GOTO 586
0
5840 NEXT E
5860 LET E=E-1
5870 LET T(26)=INT ((E(E)+(T(25)
-D(E))*F(E)/100)+.5)
5890 LET B=0
5895 CLS
5900 GOTO 500
5999 REM PRINT RESULTS
6000 CLS
6010 PRINT V$;TAB 20;Y$;TAB 27;Z
$
6015 PRINT X$
6020 FOR I=1 TO 13
6022 PRINT D$(I);
6024 LET V=S(I)
6026 LET C=24
6028 GOSUB 9500
6030 LET V=T(I)
6032 LET C=31
6034 GOSUB 9500
6035 NEXT I
6040 PRINT X$
6042 PRINT D$(14);
6044 LET V=S(14)
6046 LET C=24
6048 GOSUB 9500
6050 LET V=T(14)
6051 LET C=31
6052 GOSUB 9500
6053 PRINT AT 21,0;M$
6054 LET Z=USR 8198
6055 SLOW
6056 PAUSE 4E4
6059 CLS
6060 PRINT W$;TAB 20;Y$;TAB 27;Z
$
6062 PRINT X$
6064 FOR I=15 TO 19
6066 PRINT D$(I);
6068 LET V=S(I)
6070 LET C=24
6072 GOSUB 9500
6074 LET V=T(I)
6076 LET C=31
6078 GOSUB 9500
6080 NEXT I
6100 PRINT X$
6110 FOR I=20 TO 26
6120 PRINT D$(I);
6130 LET V=S(I)
6140 LET C=24
6150 GOSUB 9500
6160 LET V=T(I)
6170 LET C=31
6180 GOSUB 9500
6185 PRINT X$
6200 NEXT I
6300 LET Z=USR (8198)
6310 SLOW
6894 PRINT M$
6895 PAUSE 4E4
6896 CLS
6897 GOTO 500
6900 REM 82 TAX
6910 IF S=1 THEN GOSUB 7000
6920 IF S=2 THEN GOSUB 7200
6925 RETURN
7000 LET A(1)=2300
7001 LET A(2)=3400
7002 LET A(3)=4400
7003 LET A(4)=6500
7004 LET A(5)=8500
7005 LET A(6)=10800
7006 LET A(7)=12900
7007 LET A(8)=15000
7008 LET A(9)=18200
7009 LET A(10)=23500
7010 LET A(11)=28800
7011 LET A(12)=34100
7012 LET A(13)=41500
7013 LET B(1)=0
7014 LET B(2)=132
7015 LET B(3)=272
7016 LET B(4)=608
7017 LET B(5)=948
7018 LET B(6)=1385
7019 LET B(7)=1847
7020 LET B(8)=2330
7021 LET B(9)=3194
```

Continued Next Page

```
7022 LET B(10)=4837
7023 LET B(11)=6692
7024 LET B(12)=8812
7025 LET B(13)=12068
7026 LET C(1)=12
7027 LET C(2)=14
7028 LET C(3)=16
7029 LET C(4)=17
7030 LET C(5)=19
7031 LET C(6)=22
7032 LET C(7)=23
7033 LET C(8)=27
7034 LET C(9)=31
7035 LET C(10)=35
7036 LET C(11)=40
7037 LET C(12)=44
7038 LET C(13)=50
7199 RETURN
7200 LET A(1)=3400
7201 LET A(2)=5500
7203 LET A(3)=7600
7204 LET A(4)=11900
7205 LET A(5)=16000
7206 LET A(6)=20200
7207 LET A(7)=24600
7208 LET A(8)=29900
7209 LET A(9)=35200
7210 LET A(10)=45800
7211 LET A(11)=60000
7212 LET A(12)=85600
7213 LET B(1)=0
7214 LET B(2)=252
7215 LET B(3)=546
7216 LET B(4)=1234
7217 LET B(5)=2013
7218 LET B(6)=2937
7219 LET B(7)=4037
7220 LET B(8)=5574
7221 LET B(9)=7323
7222 LET B(10)=11457
7223 LET B(11)=17705
7224 LET B(12)=30249
7225 LET C(1)=12
7226 LET C(2)=14
7227 LET C(3)=16
7228 LET C(4)=19
7229 LET C(5)=22
7230 LET C(6)=25
7231 LET C(7)=29
7232 LET C(8)=33
7233 LET C(9)=39
7234 LET C(10)=44
7235 LET C(11)=49
7236 LET C(12)=50
7299 RETURN
7400 REM SETTAX
7410 IF S=1 THEN GOSUB 7500
7420 IF S=2 THEN GOSUB 7600
7425 RETURN
7500 LET D(1)=2300
7501 LET D(2)=3400
7502 LET D(3)=4400
7503 LET D(4)=8500
7504 LET D(5)=10800
7505 LET D(6)=12900
7506 LET D(7)=15000
7507 LET D(8)=18200
7508 LET D(9)=23500
7509 LET D(10)=28800
7510 LET D(11)=34100
7511 LET D(12)=41500
7512 LET D(13)=55300
7513 LET E(1)=0
7514 LET E(2)=121
7515 LET E(3)=251
7516 LET E(4)=866
7517 LET E(5)=1257
7518 LET E(6)=1656
7519 LET E(7)=2097
7520 LET E(8)=2865
7521 LET E(9)=4349
7522 LET E(10)=6045
7523 LET E(11)=7953
7524 LET E(12)=10913
7526 LET E(13)=17123
7527 LET F(1)=11
7528 LET F(2)=13
7529 LET F(3)=15
7530 LET F(4)=17
7531 LET F(5)=19
7532 LET F(6)=21
7533 LET F(7)=24
7534 LET F(8)=28
7535 LET F(9)=32
7536 LET F(10)=36
7537 LET F(11)=40
7538 LET F(12)=45
7539 LET F(13)=50
7598 RETURN
7600 LET D(1)=3400
7601 LET D(2)=5500
7602 LET D(3)=7600
7603 LET D(4)=11900
7604 LET D(5)=16000
7605 LET D(6)=20200
7606 LET D(7)=24600
7607 LET D(8)=29900
7608 LET D(9)=35200
7609 LET D(10)=45800
7610 LET D(11)=60000
7611 LET D(12)=85600
7612 LET D(13)=109400
7613 LET E(1)=0
7614 LET E(2)=231
7615 LET E(3)=504
7616 LET E(4)=1149
7617 LET E(5)=1846
7618 LET E(6)=2644
7619 LET E(7)=3656
7620 LET E(8)=5034
7621 LET E(9)=6624
7622 LET E(10)=10334
7623 LET E(11)=16014
7624 LET E(12)=27278
7625 LET E(13)=38702
7626 LET F(1)=11
7627 LET F(2)=13
7628 LET F(3)=15
7629 LET F(4)=17
7630 LET F(5)=19
7631 LET F(6)=23
7632 LET F(7)=26
7633 LET F(8)=30
7634 LET F(9)=35
7635 LET F(10)=40
7636 LET F(11)=44
7637 LET F(12)=48
7638 LET F(13)=50
7998 RETURN
7999 STOP
9500 REM JUSTIFY
9510 LET XL=INT (ABS V+.005)*SGN
  V
9540 LET J$=STR$ XL
9550 PRINT TAB (C-LEN J$+1);J$;
9560 RETURN
SYNTACTIC SUM:  58730, 8K ROM
```

by Dennis M. Filangeri, New York, NY

8K ROM/2K or 16K RAM

Spelling Mastery through Sinclair

When I bought a Timex Sinclair 1000 for my children they wanted to know if they could do their homework on it. Having a ZX81 of my own I knew it could certainly be used as a calculator, but my elementary school children would not be taken with that. So I set out to find wider applications better suited to their needs.

I decided to start them off with a 16K program that helped them practice spelling. I modified the program to fit 2K, but this limits it severely. Listing 1 gives the 16K program. Listing 2 is the 2K modified version.

After entering either version of SPELLING BEE you're ready to go. To get more memory, delete lines 1-9 in listing 2. RUN starts both versions, but Listing 1 starts automatically after LOAD once you SAVE it.

RAM size determines the number of words each version can handle. The 2K version can handle words totalling about 50 characters. The 16K version has performed without fail handling 300 characters (30 ten-character words) but may accept more. We calculate that with 16K RAM you can fit about 2000 characters. To count how many characters your word list uses, add the number of letters in each word plus one extra for each word.—Ed.)

Program Features

Each program lets you supply your own list of words. As you enter each word, the program asks you to check that you entered it correctly. You can change an incorrect entry before practice begins. In the 2K version you must start over if you enter a word incorrectly. Press ENTER when you finish entering words.

In the practice mode, either program flashes a randomly selected word from its list for a short time, then waits for the student to spell the word from memory. If he or she spells it correctly, the program flashes CORRECT and selects a different word until it exhausts the list.

If the user misspells the word, the program flashes the same word again for a slightly longer duration then loops back to wait for the user to retype the word from memory. The program does not advance to the next word until the user spells it correctly.

After completing the list the screen clears and the program displays the student's score. It tells the number of words and the number of tries at spelling them correctly. Either program then asks the user to tell it what to do next. This is where the similarities end.

16K Modes

The 16K version is menu-driven. It initially asks the user for his or her name. Figure 1 shows the menu. This program is fail-safe. It starts after loading to avoid wiping out a previously stored listing of words. If the list contains no words, the program automatically prompts the user to make a list. After completion of the list the program automatically returns to the menu in Figure 1. In mode 1 (see Figure 2), the program lets the user add to the existing list, make a new list (and thereby erase any existing list), or see the existing list. After displaying a list the program returns to mode 1. After changing the list the program returns to the main menu (Figure 1). Mode 2 is a study

```
HELLO ANN,
HERE IS A LIST OF THINGS I DO


        1) MAKE A NEW WORD LIST
        2) LET YOU STUDY THE LIST
        3) PRACTICE SPELLING DRILL
        4) SPELLING QUIZ
        5) SAVE LIST
        6) END GAME OR NEXT PLAYER


TYPE THE NUMBER AND <ENTER> TO
BEGIN
```

Figure 1.

mode. It asks the user to choose between studying the words one at a time, or studying them in a list of scrolls. The user can stop and restart scrolling. After completing the study session the program returns to the main menu.

Mode 3 (Figure 3) is a practice session as previously described. However, in the 16K version, the user gets a choice of slow, medium, or fast prompts to begin with. The prompt time increases when the user misspells a word but returns to the selected time when he or she enters the word correctly.

Mode 4 is similar to mode 3 except:

A. The prompts are always fast.

B. When the user makes an error the program goes to the next word.

```
WE WILL PRACTICE SPELLING THE
WORDS ON THE LIST. I WILL FLASH
EACH WORD, WHEN YOU SPELL IT
CORRECTLY I WILL FLASH THE NEXT
WORD. DO YOU WANT A



      1) SLOW
      2) MEDIUM
      3) FAST       GAME?




TYPE NUMBER AND <ENTER>
```

Figure 2.

```
ANN,
THERE ARE 10 WORDS ON THE LIST

DO YOU WANT TO:




       1) ADD TO THE LIST
       2) MAKE A NEW LIST
       3) SEE THE LIST




  TYPE NUMBER AND <ENTER>
```

Figure 3.

C. The program displays the misspelled words after the score. (The user can control the scrolling as in the list procedure in modes 1 and 3).

Mode 5 lets the user save his or her list to use again or to add to at a future time. The SAVE feature also has a built-in pause to allow the recorder to come up to speed, increasing the chances of a successful SAVE. Mode 6 tells the user he or she did well and prepares the program for the next user.

Entering The Program

Enter the program as shown in Listing 1 or Listing 2. I used tokens in the text when possible to save space. If you enter the program and the Syntactic Sums don't agree, DON'T PANIC. You probably spelled out a word instead of using the token (for example, spelling out F-O-R instead of using the token FOR over the F key). Save the program and then type RUN. It should bootup properly. If it crashes at least you have most of it on tape.

When typing in large amounts of text, I found it helpful to fill out the line with spaces after a PRINT statement so the first letter of text appears on the next line. This aids in putting 32 characters per line and locating where to put breaks between words. I later removed the spaces by entering the line and recalling it by EDIT. This puts the cursor at the front of the line where the spaces are. For example: in typing line 3020, I first typed:

```
3020 PRINT#####################"

WE WILL PRACTICE SPELLING THE### ...
```

then later deleted the extra spaces after PRINT.

You can change the prompt duration by altering the value of the TIME in lines 3040 and 4040 of Listing 1, or the value of D$ in line 20 of Listing 2. Change the maximum duration of error prompts by changing the value of TIME in line 3160 of Listing 1, or changing the value of T1 in line 530 of Listing 2.

Program Variables

Listing 1

C$ contains the word currently being tested.
E$ (32 spaces) erases portions of the display.
I$ stores user inputs.
N$ contains the player's name.
S$ holds the words being used and changes during play.
T$ contains the master list of words which remains constant unless the list is added to or replaced.
X$ contains the misspelled words in the quiz mode.
I accepts user inputs
J,K are loop control values.
N contains the number of words in T$.
N1 contains the number of words in S$.
W is position of word in S$ to be displayed at random.
SCORE tracks number of words correctly spelled.
TIMER is duration of prompting loop set by program.
TRIES counts number of attempts at spelling.

Listing 2

C$, E$, S$, T$ and I$ as in Listing 1.
D$ is the timer loop value held as one byte.
I, J, K, N, N1, and W as in Listing 1.
S same as SCORE.
T same as TRIES.
If you type these programs in the FAST mode, be sure to return to the SLOW mode to RUN.

Reverse Video for 16K Version

Line	
1	SPELLING BEE
260	END
380	CORRECT
420	SCORE
2120	END
2250	END
3120	END
4120	END
5020	SPELLING BEE
5040	SPELLING BEE
6050	SPELLING BEE

Reverse Video for 2K Version

Line	
490	CORRECT
630	GAME OVER

SQ

Listing 1.

```
   1 REM        SPELLING BEE
   2 REM 1982  DM FILANGERI
   3 REM        8K/16K
   4 REM
   5 REM
  10 REM BOOTUP
  20 LET E$="

  25 LET N=0
  30 RAND 0
  35 LET I=6050/1000
  40 GOTO 7050
  50 DIM S$(LEN T$)
  60 LET S$=T$
  70 LET N1=N
  75 GOSUB 145
  80 RETURN
  90 LET SCORE=-1
 100 LET TRIES=-1
 110 RETURN
 120 LET TIMER=TIME
 130 LET SCORE=SCORE+1
 140 LET TRIES=TRIES+1
 145 LET W=INT (RND*N1)
 150 RETURN
 160 LET K=0
 170 FOR J=1 TO LEN S$
 180 IF K=W THEN GOTO 220
 190 IF CODE S$(J)<>12 THEN NEXT
J
 200 LET K=K+1
 210 GOTO 180
 220 IF J>1 THEN LET J=J+1
 230 FOR K=J TO LEN S$
 240 IF CODE S$(K)<>12 THEN NEXT
K
 250 LET C$=S$(J TO K-1)
 260 IF CODE C$=0 THEN LET C$="E
ND"
 270 IF J=1 THEN LET S$=S$(K+1 T
O )
 280 IF J>1 THEN LET S$=S$( TO J
-1)+S$(K+1 TO )
 290 LET N1=N1-1
 300 RETURN
 310 PRINT AT 11,0;E$;AT 11,(32-
LEN C$)/2;C$
 320 RETURN
 330 FOR J=1 TO TIMER
 340 NEXT J
 350 PRINT AT 11,0;E$
 360 RETURN
 370 FOR J=1 TO 5
 380 PRINT AT 11,12;"CORRECT";AT
 11,12;"CORRECT";AT 11,0;E$
 390 NEXT J
 400 RETURN
 410 CLS
 420 PRINT AT 4,13;"SCORE";AT 7,
5;"WORDS",SCORE;TAB 5;"TRIES",TR
IES;AT 12,0;"NOT BAD ";N$;". PRE
SS <ENTER>"
 430 INPUT I$
 440 RETURN
1000 REM MAKE NEW LIST
1010 CLS
1015 SLOW
1020 PRINT N$;",";TAB 0;"THERE A
RE ";N;" WORDS ON THE LIST",,TAB
 0;"DO YOU WANT TO:";AT 10,5;"1)
 ADD TO THE LIST";TAB 5;"2) MAKE
 A NEW LIST";TAB 5;"3) SEE THE L
IST";AT 21,0;"TYPE NUMBER AND <E
NTER>"
1030 INPUT I
1040 IF I<1 OR I>3 THEN GOTO 100
0
1045 IF N=0 THEN LET I=2
1050 CLS
1060 GOTO 1130-20*I
1070 GOSUB 2140
1080 GOTO 1000
1090 LET T$=""
1100 LET N=0
1110 PRINT "TYPE WORDS YOU WANT
ADDED TO THELIST. JUST TYPE <ENT
ER> WHEN      FINISHED"
1120 PRINT AT 11,0;E$
1130 INPUT I$
1140 IF I$="" THEN RETURN
1150 PRINT AT 11,0;E$;E$;AT 11,0
;I$,"CORRECT?(Y/N)";AT 21,0;E$
1160 PAUSE 40000
1170 IF INKEY$<>"Y" THEN GOTO 11
20
1180 PRINT AT 11,16;E$;AT 21,0;"
NEXT WORD"
1190 LET T$=I$+"£"+T$
1200 LET N=N+1
1210 GOTO 1130
2000 REM STUDY LISTING
2010 CLS
2020 PRINT "YOU CAN STUDY THE WO
RDS:";AT 8,5;"1) ONE AT A TIME";
TAB 5;"2) IN A LIST";AT 21,0;"TY
PE NUMBER AND <ENTER>"
```

Continued Next Page

```
2030 INPUT I
2040 IF I<1 OR I>2 THEN GOTO 200
0
2050 CLS
2060 GOTO 2000+70*I
2070 PRINT "EACH WORD FROM THE L
IST WILL     APPEAR ONE AT A TIME
 IN THE      CENTER OF THE SCREEN
.";AT 21,0;"PRESS <ENTER> FOR NE
XT WORD"
2075 PAUSE 40000
2080 GOSUB 50
2090 GOSUB 145
2095 GOSUB 160
2100 GOSUB 310
2110 INPUT I$
2120 IF C$<>"END" THEN GOTO 2090
2130 RETURN
2140 PRINT AT 11,0;"EACH WORD FR
OM THE LIST WILL     APPEAR AT TH
E BOTTOM OF THE      SCREEN. PRES
S ANY KEY TO STOP    AND START TH
E LIST. PRESS TO GO"
2150 PAUSE 40000
2160 FOR J=1 TO 10
2170 SCROLL
2180 NEXT J
2190 GOSUB 50
2200 GOSUB 145
2205 GOSUB 160
2210 SCROLL
2220 PRINT TAB (32-LEN C$)/2;C$
2225 SCROLL
2230 IF INKEY$="" THEN GOTO 2250
2240 PAUSE 40000
2250 IF C$<>"END" THEN GOTO 2200
2260 FOR J=1 TO 21
2270 IF INKEY$<>"" THEN GOTO 230
0
2271 FOR K=1 TO 3
2272 NEXT K
2280 SCROLL
2290 NEXT J
2300 FAST
2310 RETURN
3000 REM PRACTICE SPELLING DRILL
3010 CLS
3020 PRINT "WE WILL PRACTICE SPE
LLING THE    WORDS ON THE LIST. I
 WILL FLASH EACH WORD, WHEN YOU
SPELL IT     CORRECTLY I WILL FLA
SH THE NEXT WORD. DO YOU WANT A"
;AT 10,5;"1) SLOW";TAB 5;"2) MED
IUM";TAB 5;"3) FAST","GAME?";AT
21,0;"TYPE NUMBER AND <ENTER>"
3030 INPUT I
3040 LET TIME=32-8*I
3050 GOSUB 50
3060 GOSUB 90
3070 GOSUB 120
3080 CLS
3090 GOSUB 160
3100 GOSUB 310
3110 GOSUB 330
3120 IF C$="END" THEN GOTO 3210
3130 INPUT I$
3140 IF I$=C$ THEN GOTO 3180
3150 GOSUB 140
3160 IF TIMER<=72 THEN LET TIMER
=TIMER+8
3170 GOTO 3100
3180 GOSUB 370
3190 GOSUB 120
3200 GOTO 3090
3210 GOSUB 420
3220 RETURN
4000 REM SPELLING QUIZ
4005 LET X$=""
4007 LET N2=0
4010 CLS
4020 PRINT AT 7,0;"YOU WILL HAVE
 ONE CHANCE AT EACHWORD. AFTER T
HE LAST WORD YOUR  SCORE WILL BE
 DISPLAYED.";TAB 11;"GOOD LUCK."
;TAB 5;"PRESS <ENTER> TO BEGIN"
4030 INPUT I$
4040 LET TIME=8
4050 GOSUB 50
4060 GOSUB 90
4070 GOSUB 120
4080 CLS
4090 GOSUB 160
4100 GOSUB 310
4110 GOSUB 330
4120 IF C$="END" THEN GOTO 4200
4130 INPUT I$
4140 IF I$=C$ THEN GOTO 4170
4145 LET X$=C$+"£"+X$
4147 LET N2=N2+1
4150 GOSUB 140
4160 GOTO 4090
4170 GOSUB 370
4180 GOSUB 130
4190 GOTO 4090
4200 GOSUB 420
4205 IF X$="" THEN RETURN
4210 DIM S$(LEN X$)
4220 LET S$=X$
4225 LET N1=N2
4230 CLS
4240 PRINT AT 7,0;"THESE ARE THE
 WORDS YOU MISSED. THEY WILL APP
EAR AT THE BOTTOM  OF THE SCREEN
."
4250 GOSUB 2200
4260 RETURN
5000 REM SAVE
5010 CLS
5020 PRINT AT 11,2;"PREPARE TO S
AVE SPELLING BEE";TAB 0;"START R
ECORDER-""CONT"" WHEN READY"
5030 STOP
5035 CLS
5040 PRINT AT 11,0;"PROGRAM SAVI
NG SPELLING BEE"
5045 PAUSE 120
5050 SAVE "SPELLING BEE"
5055 LET I=6
5060 GOTO 7050
6000 REM EXIT GAME
6010 CLS
6020 PRINT AT 5,0;"WELL, ";N$;",
 YOU DID FINE.";TAB 0;"I HOPE YO
U HAD AS MUCH FUN AS    I DID. PR
ESS <ENTER> TO LET    SOMEONE E
LSE HAVE A CHANCE.";TAB 0;"HAVE
A NICE DAY"
6030 INPUT I$
6040 CLS
6050 PRINT AT 10,10;"WE WILL PLA
Y";TAB 10;"SPELLING BEE";TAB 5;"
PLEASE TYPE YOUR NAME"
6060 INPUT N$
7000 REM MENU
7010 CLS
```

Continued Next Page

```
7020 PRINT "HELLO ";N$;",";TAB 0
;"HERE IS A LIST OF THINGS I DO"
;AT 5,5;"1) MAKE A NEW WORD LIST
";TAB 5;"2) LET YOU STUDY THE LI
ST";TAB 5;"3) PRACTICE SPELLING
DRILL";TAB 5;"4) SPELLING QUIZ";
TAB 5;"5) SAVE LIST ";TAB 5;"6)
END GAME OR NEXT PLAYER",,,,TAB
 0;"TYPE THE NUMBER AND <ENTER>
TO  BEGIN"
7030 SLOW
7040 INPUT I
7050 IF I<1 OR I>6 THEN GOTO 700
0
7055 IF N=0 THEN LET I=1
7060 GOSUB 1000*I
7070 GOTO 7000
SYNTACTIC SUM:  27925, 8K ROM
```

Listing 2.

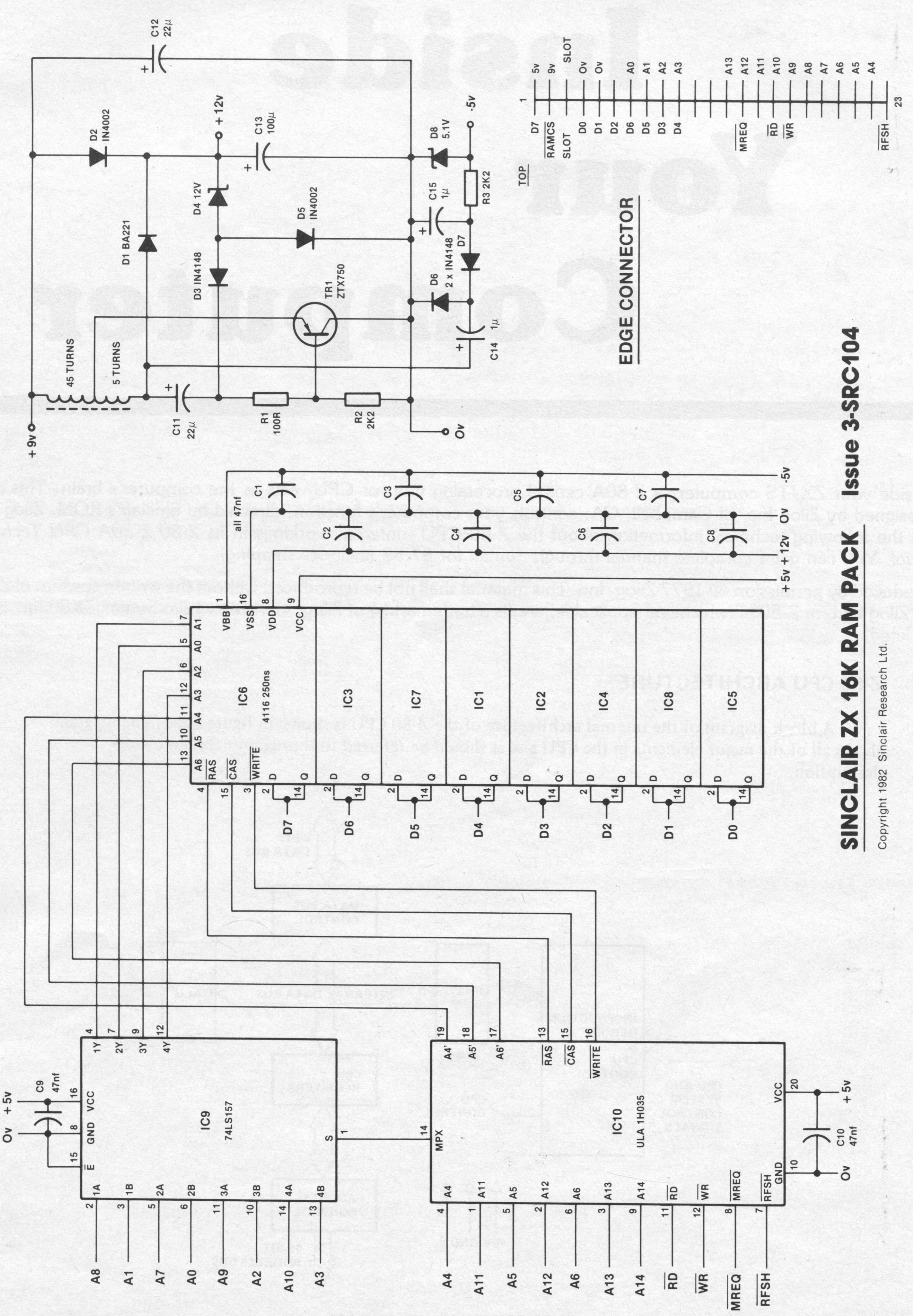
```
   1 REM SPELLING BEE
   2 REM 8K/2K
   3 REM 1982 D.M. FILANGERI
   4 REM DELETE LINES 1 TO 9
   5 REM
   6 REM
   7 REM
   8 REM
   9 REM
  10 LET T$=""
  20 LET D$=CHR$ 8
  30 DIM E$(32)
  40 CLS
  50 PRINT "TYPE WORDS"
  60 LET N=0
  70 PRINT AT 1,0;E$
  80 INPUT I$
  90 IF I$="" THEN GOTO 170
 100 PRINT AT 1,0;E$;E$;AT 1,0;I
$,"CORRECT?(Y/N)",E$
 110 PAUSE 40000
 120 IF INKEY$<>"Y" THEN GOTO 70
 130 PRINT AT 1,16;E$;AT 2,0;"NE
XT WORD"
 140 LET T$=I$+"£"+T$
 150 LET N=N+1
 160 GOTO 80
 170 CLS
 180 LET S=0
 190 LET T=0
 200 LET N1=N
 210 DIM S$(LEN T$)
 220 LET S$=T$
 230 LET W=INT (RND*N1)
 240 LET T1=CODE D$
 250 LET K=0
 260 FOR J=1 TO LEN S$
 270 IF K=W THEN GOTO 310
 280 IF CODE S$(J)<>12 THEN NEXT
J
 290 LET K=K+1
 300 GOTO 270
 310 IF J>1 THEN LET J=J+1
 320 FOR K=J TO LEN S$
 330 IF CODE S$(K)<>12 THEN NEXT
K
 340 LET C$=S$(J TO K-1)
 350 IF CODE C$=0 THEN GOTO 560
 360 LET S=S+1
 370 LET T=T+1
 380 IF J=1 THEN LET S$=S$(K+1 T
O )
 390 IF J>1 THEN LET S$=S$( TO J
-1)+S$(K+1 TO )
 400 LET N1=N1-1
 410 CLS
 420 PRINT C$
 430 FOR J=1 TO T1
 440 NEXT J
 450 CLS
 460 INPUT I$
 470 IF I$<>C$ THEN GOTO 530
 480 FOR J=1 TO 5
 490 PRINT AT 0,0;"CORRECT";AT 0
,0;"CORRECT"
 500 NEXT J
 510 CLS
 520 GOTO 230
 530 IF T1<=94 THEN LET T1=T1+6
 540 LET T=T+1
 550 GOTO 410
 560 CLS
 570 PRINT "WORDS",S,"TRIES",T,"
PLAY AGAIN?",,"1)SAME WORDS, 2)N
EW, 3)NO"
 580 INPUT I$
 590 IF I$="1" THEN GOTO 170
 600 IF I$="2" THEN GOTO 10
 610 IF I$<>"3" THEN GOTO 580
 620 CLS
 630 PRINT AT 11,11;"GAME OVER"
SYNTACTIC SUM:  15301,8K ROM
```


PEGASUS MICRO SYSTEMS

SOFTWARE & HARDWARE ENGINEERS
P.O. BOX 397
CHESTERTOWN, MD 21620
Phone 301-348-5865

	ZX81 & TS1000 or Timex 16K	ZX81 or TS1000 (48K +)	Timex 2000 or Spectrum (48K)
(1) Astro Map & Conversion	$9.95	Expanded $14.95	Hi. Res. Vers $14.95
(2) Design List & Other List Programs	$9.95	Expanded $14.95	Hi. Res. Vers. $14.95
(3) Savings & Loan Advis. (Amort. & Inv. Counselor)	$9.95	Expanded $14.95	Hi. Res. Vers. $14.95
(4) Music Anth.	Not Avail.	Not Avail.	$14.95
(5) Artist's Friend	Not Avail.	Not Avail.	$14.95
(6) Cycloids	Not Avail.	Not Avail.	$14.95
(7) Bio-map	Not Avail.	Not Avail.	$14.95
(8) Omnigraph	Not Avail.	Not Avail.	$14.95

Attention software engineers: If you think you have a great program and want the best return on your effort, contact Pegasus Micro Systems at above address or call 301-348-5865. We offer 80% of net profits for work done by you only; 60% for your work which we help with, and 50% if you head up a team effort. Write or call today. Remember, we also help programmers with their problems as we are programmers!


SINCLAIR ZX 16K RAM PACK Issue 3-SRC104
Copyright 1982 Sinclair Research Ltd.
EDGE CONNECTOR
TOP
D7 RAMCS SLOT D0 D1 D2 D6 D5 D3 D4 MREQ RD WR RFSH
5v 9v SLOT 0v 0v A0 A1 A2 A3 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4
+9v
45 TURNS
5 TURNS
D2 IN4002
D1 BA221
D3 IN4148
D4 12V
D5 IN4002
TR1 ZTX750
D6 2 x IN4148 D7
D8 5.1V
C11 22µ
C12 22µ
C13 100µ
C14 1µ
C15 1µ
R1 100R
R2 2K2
R3 2K2
+12v
-5v
0v
all 47nf
C1 C2 C3 C4 C5 C6 C7 C8
+5v +12v 0v -5v
VBB VSS VDD VCC
A1 A0 A2 A3 A4 A5 A6 RAS CAS WRITE D Q
IC6 4116 250ns
IC3 IC7 IC1 IC2 IC4 IC8 IC5
D7 D6 D5 D4 D3 D2 D1 D0
IC9 74LS157
GND VCC 1Y 2Y 3Y 4Y E S 1A 1B 2A 2B 3A 3B 4A 4B
C9 47nf
0v +5v
A8 A1 A7 A0 A9 A2 A10 A3
IC10 ULA 1H035
A4' A5' A6' RAS CAS WRITE MPX VCC GND
A4 A11 A5 A12 A6 A13 A14 RD WR MREQ RFSH
C10 47nf
0v +5v

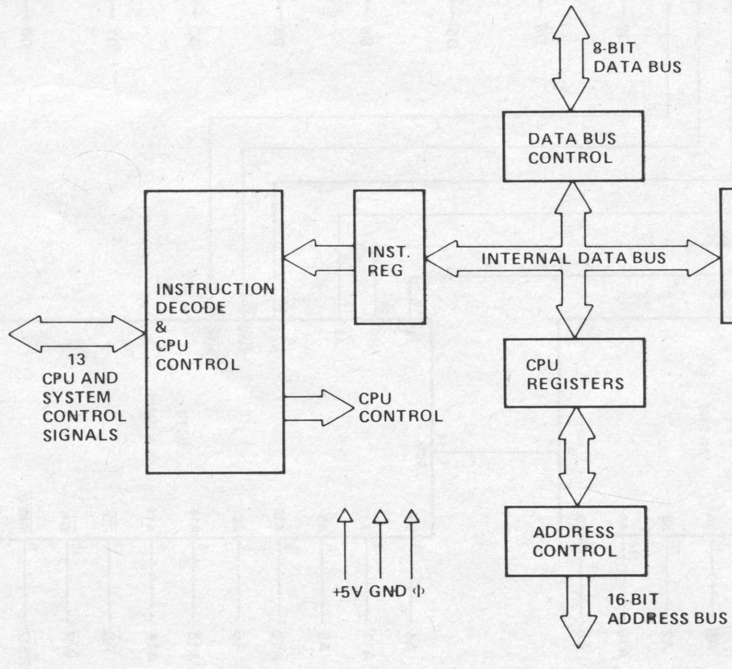
Inside Your Computer

Inside your ZX/TS computer, a Z-80A central processing unit, or CPU, acts as the computer's brain. This chip, designed by Zilog Inc. of Campbell, CA, controls your computer's function, directed by Sinclair's ROM. Zilog provides the following technical information about the Z-80 CPU's internal workings in its *Z-80 Z-80A CPU Technical Manual*. You can get a complete manual through Syntax for $7.88 (includes shipping).

Reproduced by permission © 1977 Zilog, Inc. This material shall not be reproduced without the written consent of Zilog, Inc. "Zilog and/or Z-80®" (whichever applicable) are/is a trademark(s) of Zilog, Inc. with whom Syntax ZX80 Inc. is not associated.

Z-80 CPU ARCHITECTURE

A block diagram of the internal architecture of the Z-80 CPU is shown in figure 2.0-1. The diagram shows all of the major elements in the CPU and it should be referred to throughout the following description.



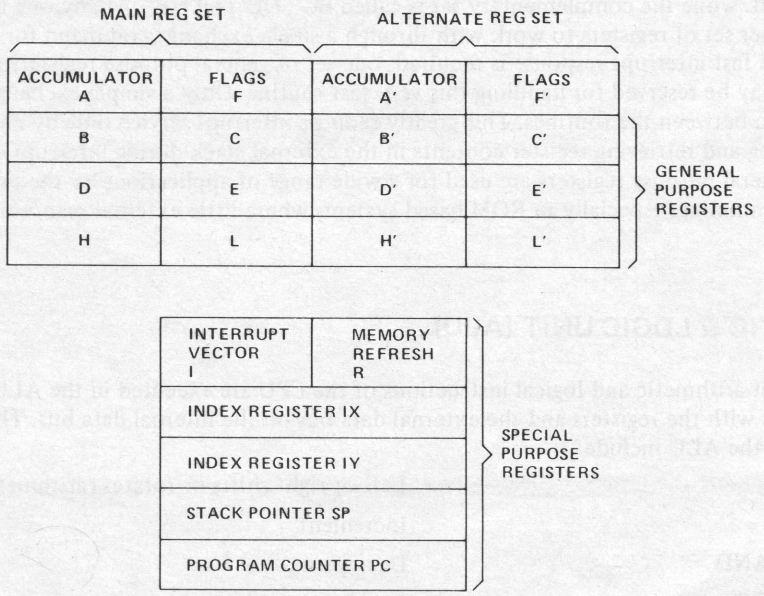
Z-80 CPU BLOCK DIAGRAM
FIGURE 2.0-1

CPU REGISTERS

The Z-80 CPU contains 208 bits of R/W memory that are accessible to the programmer. Figure 2.0-2 illustrates how this memory is configured into eighteen 8-bit registers and four 16-bit registers. All Z-80 registers are implemented using static RAM. The registers include two sets of six general purpose registers that may be used individually as 8-bit registers or in pairs as 16-bit registers. There are also two sets of accumulator and flag registers.

Special Purpose Registers

1. **Program Counter (PC).** The program counter holds the 16-bit address of the current instruction being fetched from memory. The PC is automatically incremented after its contents have been transferred to the address lines. When a program jump occurs the new value is automatically placed in the PC, overriding the incrementer.
2. **Stack Pointer (SP).** The stack pointer holds the 16-bit address of the current top of a stack located anywhere in external system RAM memory. The external stack memory is organized as a last-in first-out (LIFO) file. Data can be pushed onto the stack from specific CPU registers or popped off of the stack into specific CPU registers through the execution of PUSH and POP instructions. The data popped from the stack is always the last data pushed onto it. The stack allows simple implementation of multiple level interrupts, unlimited subroutine nesting and simplification of many types of data manipulation.



Z-80 CPU REGISTER CONFIGURATION
FIGURE 2.0-2

3. **Two Index Registers (IX & IY).** The two independent index registers hold a 16-bit base address that is used in indexed addressing modes. In this mode, an index register is used as a base to point to a region in memory from which data is to be stored or retrieved. An additional byte is included in indexed instructions to specify a displacement from this base. This displacement is specified as a two's complement signed integer. This mode of addressing greatly simplifies many types of programs, especially where tables of data are used.
4. **Interrupt Page Address Register (I).** The Z-80 CPU can be operated in a mode where an indirect call to any memory location can be achieved in response to an interrupt. The I Register is used for this purpose to store the high order 8-bits of the indirect address while the interrupting device provides the lower 8-bits of the address. This feature allows interrupt routines to be dynamically located anywhere in memory with absolute minimal access time to the routine.

5. **Memory Refresh Register (R).** The Z-80 CPU contains a memory refresh counter to enable dynamic memories to be used with the same ease as static memories. Seven bits of this 8 bit register are automatically incremented after each instruction fetch. The eighth bit will remain as programmed as the result of an LD R, A instruction. The data in the refresh counter is sent out on the lower portion of the address bus along with a refresh control signal while the CPU is decoding and executing the fetched instruction. This mode of refresh is totally transparent to the programmer and does not slow down the CPU operation. The programmer can load the R register for testing purposes, but this register is normally not used by the programmer. During refresh, the contents of the I register are placed on the upper 8 bits of the address bus.

Accumulator and Flag Registers

The CPU includes two independent 8-bit accumulators and associated 8-bit flag registers. The accumulator holds the results of 8-bit arithmetic or logical operations while the flag register indicates specific conditions for 8 or 16-bit operations, such as indicating whether or not the result of an operation is equal to zero. The programmer selects the accumulator and flag pair that he wishes to work with with a single exchange instruction so that he may easily work with either pair.

General Purpose Registers

There are two matched sets of general purpose registers, each set containing six 8-bit registers that may be used individually as 8-bit registers or as 16-bit register pairs by the programmer. One set is called BC, DE and HL while the complementary set is called BC′, DE′ and HL′. At any one time the programmer can select either set of registers to work with through a single exchange command for the entire set. In systems where fast interrupt response is required, one set of general purpose registers and an accumulator/flag register may be reserved for handling this very fast routine. Only a simple exchange commands need be executed to go between the routines. This greatly reduces interrupt service time by eliminating the requirement for saving and retrieving register contents in the external stack during interrupt or subroutine processing. These general purpose registers are used for a wide range of applications by the programmer. They also simplify programming, especially in ROM based systems where little external read/write memory is available.

ARITHMETIC & LOGIC UNIT (ALU)

The 8-bit arithmetic and logical instructions of the CPU are executed in the ALU. Internally the ALU communicates with the registers and the external data bus on the internal data bus. The type of functions performed by the ALU include:

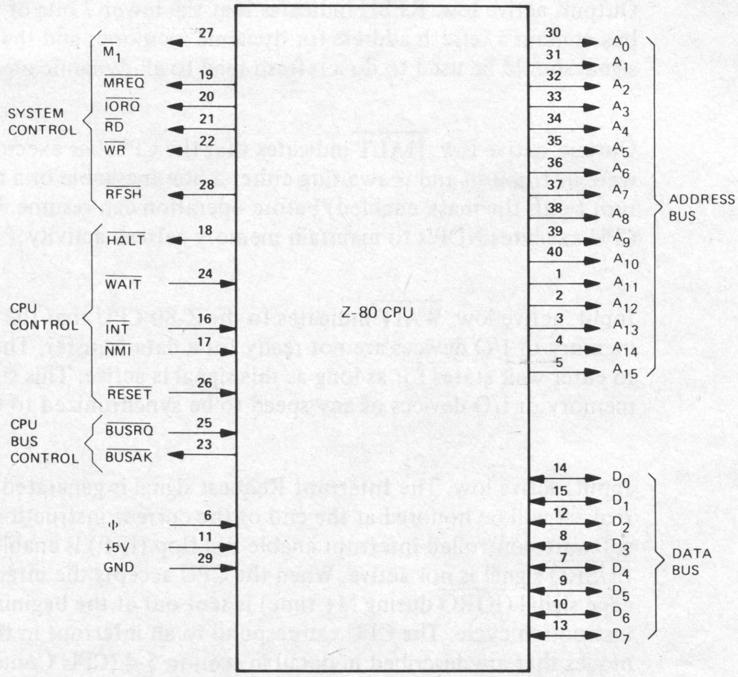
Add	Left or right shifts or rotates (arithmetic and logical)
Subtract	Increment
Logical AND	Decrement
Logical OR	Set bit
Logical Exclusive OR	Reset bit
Compare	Test bit

INSTRUCTION REGISTER AND CPU CONTROL

As each instruction is fetched from memory, it is placed in the instruction register and decoded. The control sections performs this function and then generates and supplies all of the control signals necessary to read or write data from or to the registers, control the ALU and provide all required external control signals.

Z-80 CPU PIN DESCRIPTION

The Z-80 CPU is packaged in an industry standard 40 pin Dual In-Line Package. The I/O pins are shown in figure 3.0-1 and the function of each is described below.



Z-80 PIN CONFIGURATION
FIGURE 3.0-1

$\overline{IORQ}$
(Input/Output Request)

Tri-state output, active low. The $\overline{IORQ}$ signal indicates that the lower half of the address bus holds a valid I/O address for a I/O read or write operation. An $\overline{IORQ}$ signal is also generated with an $\overline{M1}$ signal when an interrupt is being acknowledged to indicate that an interrupt response vector can be placed on the data bus. Interrupt Acknowledge operations occur during M_1 time while I/O operations never occur during M_1 time.

$\overline{RD}$
(Memory Read)

Tri-state output, active low. $\overline{RD}$ indicates that the CPU wants to read data from memory or an I/O device. The addressed I/O device or memory should use this signal to gate data onto the CPU data bus.

$\overline{WR}$
(Memory Write)

Tri-state output, active low. $\overline{WR}$ indicates that the CPU data bus holds valid data to be stored in the addressed memory or I/O device.

A_0-A_{15}
(Address Bus)

Tri-state output, active high. A_0-A_{15} constitute a 16-bit address bus. The address bus provides the address for memory (up to 64K bytes) data exchanges and for I/O device data exchanges. I/O addressing uses the 8 lower address bits to allow the user to directly select up to 256 input or 256 output ports. A_0 is the least significant address bit. During refresh time, the lower 7 bits contain a valid refresh address.

D_0-D_7
(Data Bus)

Tri-state input/output, active high. D_0-D_7 constitute an 8-bit bidirectional data bus. The data bus is used for data exchanges with memory and I/O devices.

$\overline{M_1}$
(Machine Cycle one)

Output, active low. $\overline{M_1}$ indicates that the current machine cycle is the OP code fetch cycle of an instruction execution. Note that during execution of 2-byte op-codes, $\overline{M1}$ is generated as each op code byte is fetched. These two byte op-codes always begin with CBH, DDH, EDH or FDH. $\overline{M1}$ also occurs with $\overline{IORQ}$ to indicate an interrupt acknowledge cycle.

$\overline{MREQ}$
(Memory Request)

Tri-state output, active low. The memory request signal indicates that the address bus holds a valid address for a memory read or memory write operation.

$\overline{\text{RFSH}}$ (Refresh)

Output, active low. $\overline{\text{RFSH}}$ indicates that the lower 7 bits of the address bus contain a refresh address for dynamic memories and the current $\overline{\text{MREQ}}$ signal should be used to do a refresh read to all dynamic memories.

$\overline{\text{HALT}}$ (Halt state)

Output, active low. $\overline{\text{HALT}}$ indicates that the CPU has executed a HALT software instruction and is awaiting either a non maskable or a maskable interrupt (with the mask enabled) before operation can resume. While halted, the CPU executes NOP's to maintain memory refresh activity.

$\overline{\text{WAIT}}$ (Wait)

Input, active low. $\overline{\text{WAIT}}$ indicates to the Z-80 CPU that the addressed memory or I/O devices are not ready for a data transfer. The CPU continues to enter wait states for as long as this signal is active. This signal allows memory or I/O devices of any speed to be synchronized to the CPU.

$\overline{\text{INT}}$ (Interrupt Request)

Input, active low. The Interrupt Request signal is generated by I/O devices. A request will be honored at the end of the current instruction if the internal software controlled interrupt enable flip-flop (IFF) is enabled and if the $\overline{\text{BUSRQ}}$ signal is not active. When the CPU accepts the interrupt, an acknowledge signal ($\overline{\text{IORQ}}$ during M_1 time) is sent out at the beginning of the next instruction cycle. The CPU can respond to an interrupt in three different modes that are described in detail in section 5.4 (CPU Control Instructions).

$\overline{\text{NMI}}$ (Non Maskable Interrupt)

Input, negative edge triggered. The non maskable interrupt request line has a higher priority than $\overline{\text{INT}}$ and is always recognized at the end of the current instruction, independent of the status of the interrupt enable flip-flop. $\overline{\text{NMI}}$ automatically forces the Z-80 CPU to restart to location 0066_H. The program counter is automatically saved in the external stack so that the user can return to the program that was interrupted. Note that continuous $\overline{\text{WAIT}}$ cycles can prevent the current instruction from ending, and that a $\overline{\text{BUSRQ}}$ will override a $\overline{\text{NMI}}$.

$\overline{\text{RESET}}$

Input, active low. $\overline{\text{RESET}}$ forces the program counter to zero and initializes the CPU. The CPU initialization includes:

1) Disable the interrupt enable flip-flop
2) Set Register I = 00_H
3) Set Register R = 00_H
4) Set Interrupt Mode 0

During reset time, the address bus and data bus go to a high impedance state and all control output signals go to the inactive state.

$\overline{\text{BUSRQ}}$ (Bus Request)

Input, active low. The bus request signal is used to request the CPU address bus, data bus and tri-state output control signals to go to a high impedance state so that other devices can control these buses. When $\overline{\text{BUSRQ}}$ is activated, the CPU will set these buses to a high impedance state as soon as the current CPU machine cycle is terminated.

$\overline{\text{BUSAK}}$ (Bus Acknowledge)

Output, active low. Bus acknowledge is used to indicate to the requesting device that the CPU address bus, data bus and tri-state control bus signals have been set to their high impedance state and the external device can now control these signals.

Φ

Single phase TTL level clock which requires only a 330 ohm pull-up resistor to +5 volts to meet all clock requirements.

by John Oliger, Indianapolis, IN

Build a 64K Dynamic Memory Board

In this project we will build our own 64K memory board. Why should you want to go to the trouble of building your own when commercially built units are available? For two reasons. One, you can build your own for substantially less than if you bought a prebuilt unit. You can build this memory today for less than $75 while the ready-built ones cost as much as $179! Second, you may prefer to build your own to understand the operation of your hardware better. If you follow through the construction and theory of operation of this circuit, you will begin to gain this knowledge. If you know both hardware and software for a computer, you are its true master. What you can not do one way, you can do the other way.

Also, if later your memory gives you trouble, you should really be able to service it yourself. Sometimes, the best way to service a board is by parts substitution. In many commercially available units, the chips, including the memory chips, are soldered directly to the board without sockets. Just try to remove a $5.00 memory chip from a PC board with plated-through holes. You will see how easily you can damage the board, chip, or both! Also, some manufacturers are so proud of their circuits that they won't even *sell* you a copy of a schematic. The most popular ZX/TS 64K memory supplier uses a custom programmed PROM (not EPROM) for an address decoder/logic element. Yes, it is soldered directly to the PC board. But you probably couldn't replace it even if you could get it loose without destroying the board.

Our board is designed for use with the expansion board we built in SQ Winter 1982. If you like, you could use it alone by either soldering two 46-pin edge connectors back to back or by simply soldering a 46-pin edge connector directly to the edge connector traces on the memory board and plugging it in. It should also work on any Sinclair bus-compatible expansion board you may purchase from another source. The board draws a little more than 100 mA of current from the computer, so you shouldn't encounter any new power supply problems.

My circuit is designed around the pin-1-RFSH type 4164 dynamic memory chip. I chose this chip because Sinclair used the internal Z80 refresh register in the video display routine. According to the 4164 data sheets, if we used the internal Z80 refresh register to refresh the memory, certain rows needing refresh would not be refreshed often enough. I tested this and found that in FAST mode, the worst case refresh address (DCH) is only refreshed every 18.2 ms. In SLOW mode this row is only refreshed every 13.7 ms. Specs for these dynamic memories call for a refresh to every refresh row at least every 2 ms. Hence, if we used the internal Z80 refresh register, we would run the chips to nine times out of spec!

Strangely enough, it would work just fine this way! My 16K memory board did use the internal R register and had no refresh problems, and a number of commercially available 64K memories use the R register for refresh and appear to have no problems. But we want this memory to be completely in spec, so we'll use the pin-1-refresh chip. (Evidently Sinclair thought external RAM needed it, because their 16K memory pack contains a separate refresh counter. Using the pin-1-refresh chip eliminates this hardware, while still providing "up to snuff" refresh to the memory.) The disadvantage of using this chip is that the pin-1-refresh 4164 is harder to find than the non-pin-1-refresh type. You'll find two sources of Mitsubishi 200 ns pin-1-refresh chips listed on the schematic.

I can get the 150 ns version for you at $7.00 each if you cannot locate any. This is my actual cost including shipping. I offer these only as an alternative chip source, if you cannot readily get them from the other two sources. But why pay extra? The 200 ns version available at $4.99 works fine!

If you get a Mitsubishi part and want to verify that it is the pin-1-refresh type, just look at its part number. If it says M5K4164S-xx or M5K4164P-xx, then it is the correct type. If it says M5K4164NS-xx or M5K4164NP- xx, then you have problems. The "N" in the number means "NC" on pin 1. It will **NOT** work in this circuit. Non- pin-1-refresh type 4164s are popular as upgrades from 16K because pin 1 on 4116s is connected to -5V. Because there is no connection to pin 1 on these 4164s, it eliminates some trace cutting. Thus, the non-pin-1-refresh chip is more popular—hence, more readily available. Prices of the two types are comparable.

Back to the board. This memory board fully decodes Sinclair's ROM when installed. This opens up the address space from 8-16K and from 32-48K (i.e., any address where bit 14 is low). If you do NOT wish this decoding performed because you have already done it elsewhere (possibly on one of my EPROM read boards), then you can easily disable this function. Simply bend out pin 15 of U5 (74LS138) before installing it in its socket so this pin doesn't enter the socket. ROM decoding will then be totally up to you to perform.

If you wish to disable RAM from 2000-2FFFH, or from 3000-3FFFH, for EPROM boards or memory-mapped I/Os, use the 8-12K and 12-16K on/off switches. You also have two banks of 8K RAM available in this 8-16K region. The "BANK A/B" switch selects which 8K bank is currently available for use. Therefore, for example, if you turn the 8-12K on/off switch off, you still would have two 4K banks of memory available for data storage from 12-16K. These addresses could be very handy for MC storage. Thus this circuit really provides a full 64K of memory. Some other 64K memories actually have only 56K of memory available for use (they count the ROM, too). An inboard switch can also turn off the top 16K (from 48K-64K) of memory. Why would you want to do this? Because later we'll build a 2764-based "cartridge board" that will use this block.

If you built my Printer Port board from the premier SQ, then you will be interested in the PORT SEL NOT input on this memory board. This input is tied to trace 50B on this board, so all you need to keep the memory board from fighting the printer port is to tie PORT SEL NOT on the printer port to trace 50B on its edge connector trace.

Board Construction

If you want to build your own PC board, use Layout 1 at full size. See SQ, Winter 82, for specific directions on PC board making. If you prefer to buy a board, see the ending notes for ordering directions and information.

Installing the feed-throughs consumes the most construction time for this board. You'll install almost 100 of them. Expect to spend a couple of hours just installing feed-throughs. It takes me almost exactly 1 1/2 hours to install these on this board, and I designed it! When you finish installing these feed-throughs, consider yourself over half-way finished.

Assembly Instructions

Before you start wiring, erase the board with a clean pencil eraser. It removes oxides, dirt and finger oils, reducing the chance of a poor joint.

First solder all feed-throughs onto the board. Use 30AWG solid wire, or stripped wire-wrap wire for this job. We prefer this small wire because it does not readily transfer heat from one side to the other when soldering on both sides. Thus, using a sponge (as I recommend in the expansion board project) is not necessary. The short length of the wire (about 1/16th inch) makes the wire diameter non-critical as far as the circuit is concerned.

Position the component side of the board toward you, edge connector pointing down. Then locate the component mounting pads that double as feed-throughs. You should find 5 locations—both ends of R2, and one end each of R1, R3 and R4. Cover these pads with small pieces of tape or self-stick label. (A single-hole paper punch cuts circular pieces quickly.) See Photo 1.

Do not put jumpers through any IC pads, those with the characteristic flattened oval shape: ▬ You already covered the component mounting holes. Now install feed-throughs in every hole with a circular pad showing on the component side.

You can control the solder flow easily using simple tricks. Solder with the board slanted upward and away from you. Bend the wire upward to the edge of the pad, not along the trace. Insert the whole piece of number 30 wire through the hole, leaving only enough to bend upward, then bend the wire on both sides of the board and bend the long tail tightly over the top of the board. Now the wire holds itself in place as you solder—first the component side, then the wiring side. When the joint cools, cut off the excess wire.

When you finish installing feed-throughs, six IC pads (on U2 through U6) and five resistor pads remain clear. All other pads on the component side contain feed-throughs. See Photo 2.

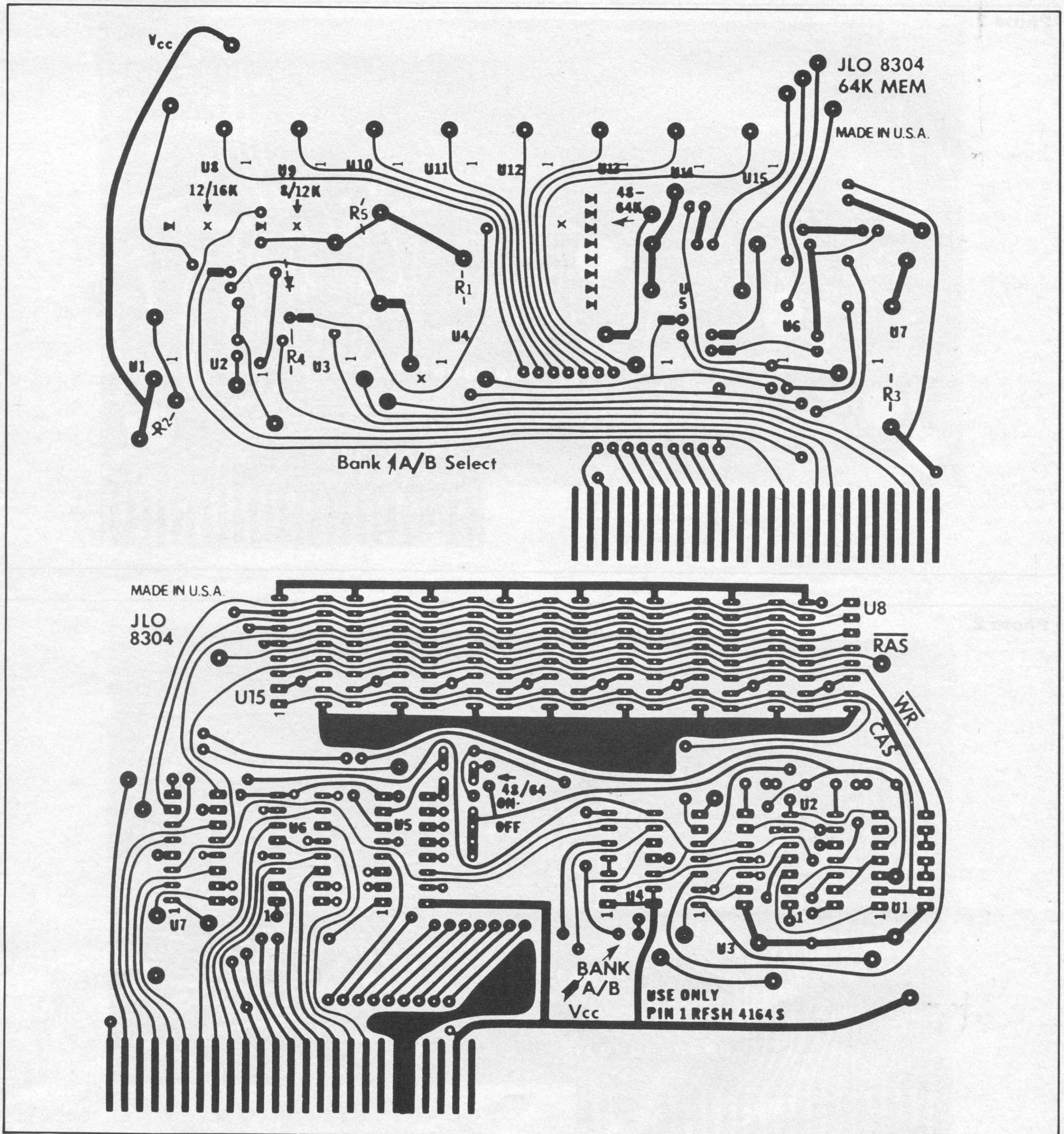
Other than the IC sockets, eleven diodes, five resistors and four switches mount on the board. Diode sites are marked on the component side of the board with standard diode symbols. Resistors are also marked on the board, as R1 to R5. The four miniature slide switches are marked on the board with an "x" between each switch's respective mounting holes. See Photo 3.

Solder all diodes and switches only to the *bottom* of the board. Except for R5, all resistors have at least one lead soldered to both sides of the board.

After soldering in all the feed-throughs, remove the cover dots from the pads and erase the board areas that got sticky. They first turn black, then clean. Next install R5, 560 Ohms. Then add the 2.7K resistors, making sure to solder all component (but not socket) leads that can be soldered on both sides. Now install the eleven diodes, making sure that the band end (cathode) of each diode is on the same end as the line in the symbol etched onto board. After you solder a few dozen connections, remove all traces of flux with acetone and a soft cloth. This improves the board's appearance and makes bad joints or shorts more readily apparent.

Next, solder all the IC sockets in place. Solder carefully where PC traces go between IC pads to avoid shorting. Now remove all flux with acetone and install the four switches. Press the switch terminals into the board as far as they will go and then solder them to the bottom. See Photo 4. (We found it helpful to bend the switch mounting tabs upward to avoid interference.—KO).

Now, install the five tantalum bypass caps by forming the leads to fit over their respective chips (three spaced out evenly on the memory array, one under U3, and one under U7). Bend the lead ends to 90 degrees, and solder them to pins 7 and 14 or 8 and 16 of the chips. Solder the red (positive) end of the cap to the positive pad.



Layout 1.

Positive pads on the logic chips are the higher pin numbers (pin 14 or pin 16), but THE MEMORY CHIPS' POWER CONNECTIONS ARE REVERSED—pin 8 is Vcc and pin 16 is ground. See Photo 5. Double check that all caps are in correctly. Several people who test-built this board found improper installation of these capacitors to be the most likely errors.

Now install the three wire jumpers. No holes are provided for these. Use a red wire from U5 pin 16 to U6 pin 16. Put two black wires from power ground (large trace area near the edge connector on bottom of board) to ground on the memory array (large trace area near memory chips on bottom of board connecting to the memory pin 16). One jumper should go to one end of the memory array; another goes to the other end.

Clean off any flux left on the board and inspect the board for shorts. Use a bright light and a magnifier.

Now, recheck that all diodes are installed in the correct direction and all bypass caps are installed with the red end connected to the positive pad.

Photo 1.

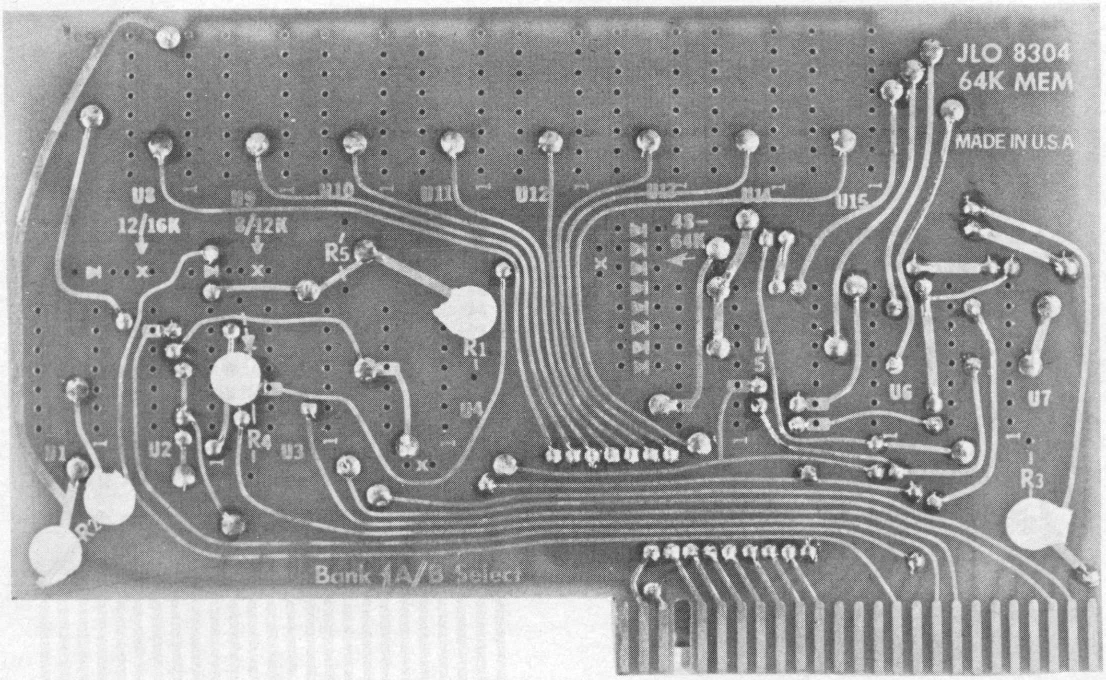
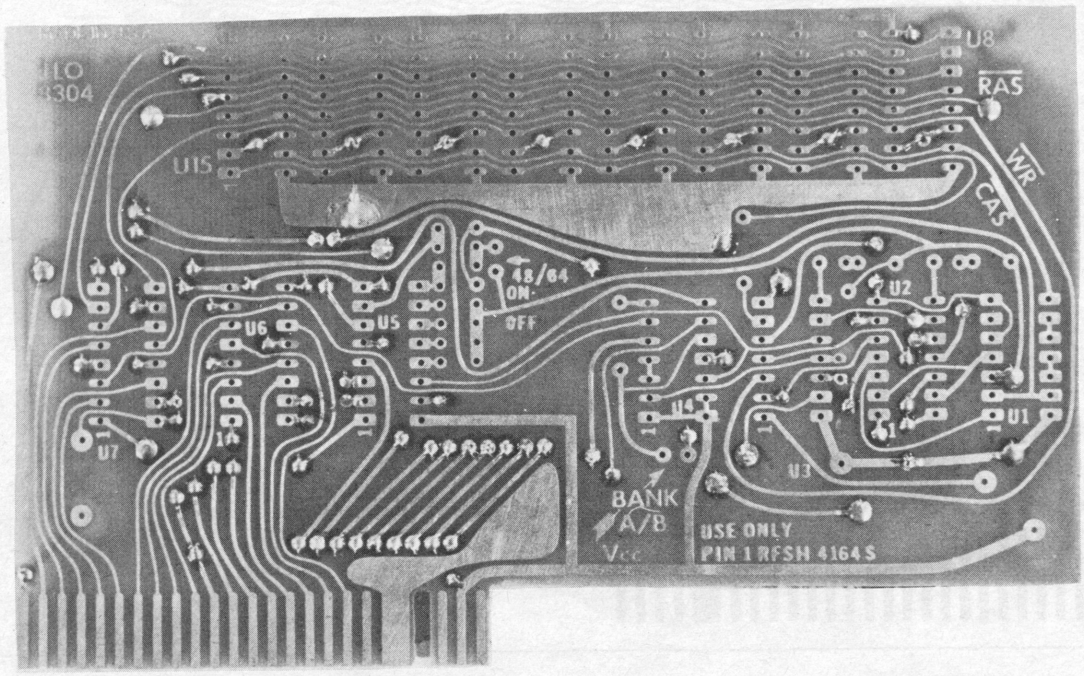


Photo 2.



CAUTION—USE ONLY PIN 1 REFRESH TYPE 4164 MEMORY CHIPS IN THIS CIRCUIT. You may use: Mitsubishi no. M5K4164S-15, M5K4164P-15, M5K4164S-20, M5K4164P-20, Mostek no. MK4164, or Motorola no. MCM 6664. Mitsubishi chips are available from: Microprocessors Unlimited, RT. 1, Box 260, Beggs, OK 74421, 918/267-4796 or 267-4242. An alternate source is Microware Exeltek, P.O. Box 5143, South San Francisco, CA 94080, 415/872-2195. Specify that you will accept only pin-1-refresh-type chips.

You can now install the ICs. Be sure to install them with pin 1 on the correct end. Pin 1 is marked on both the component and wiring side for every IC, and all ICs install with pin 1 toward the connector. Photo 6 shows the front of the finished board.

This completes the assembly.

Testing

Now we need to test our new RAM board. Insert the memory board into the expansion board slot nearest the

Photo 3.

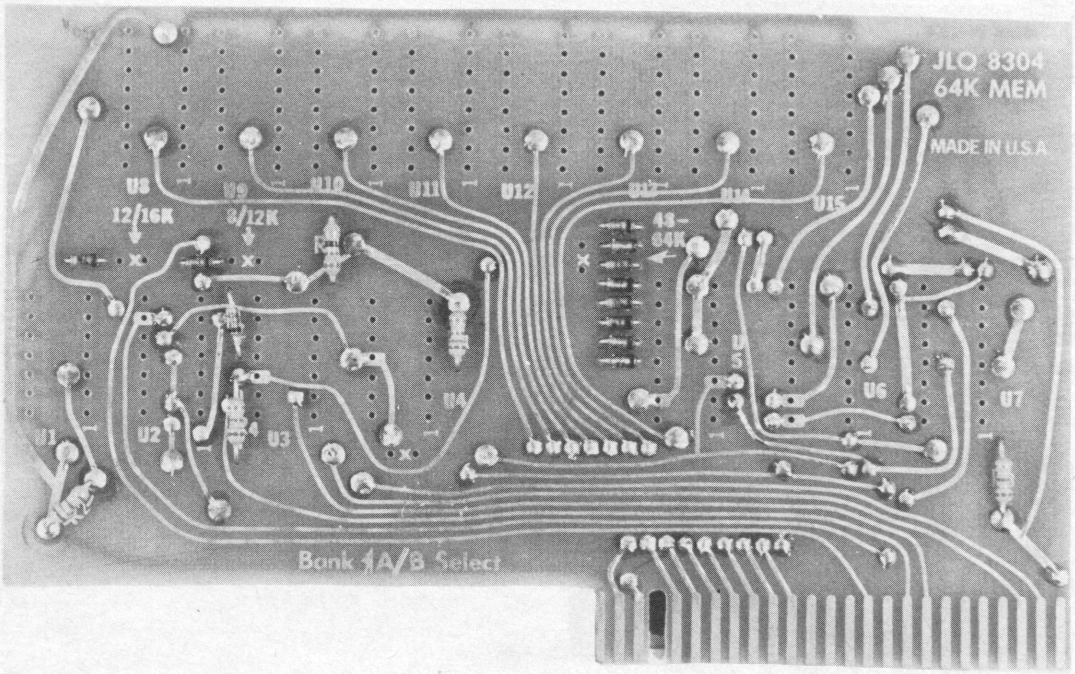
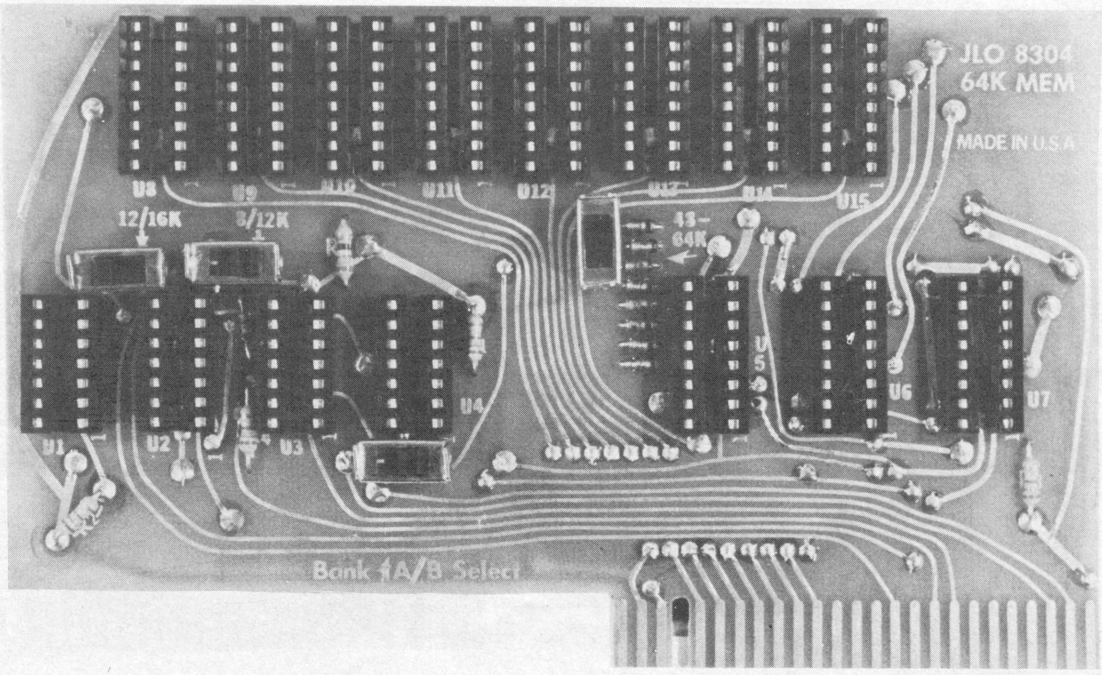


Photo 4.



computer. I like to run mine in this slot because: 1) It is easier to get to the block on/off bank-select switches. 2) The closer the board is physically to the computer, the faster the signals from the computer get to and from it. I consider this "slot" to be the expansion board's prime slot, hence the best place for the system's RAM.

Turn on the computer. After a short pause, the "K" cursor should appear as usual. If it doesn't, try turning the computer on and off a few times to see if it properly reset on power-up. If you still can't get the cursor, then remove power from the computer, remove the memory board, and start hunting for errors.

Verify that all the tantalum bypass caps on the bottom of the board are installed with the right polarity and on the correct IC pads. You can use an Ohmmeter to check continuity between all the ground ends of these caps to all the other caps' ground end. Also use the meter to check that all the positive (red) ends of the caps are connected together. If you find one that isn't, stop and find out why. Be sure the three wire jumpers are soldered to the correct

Photo 5.

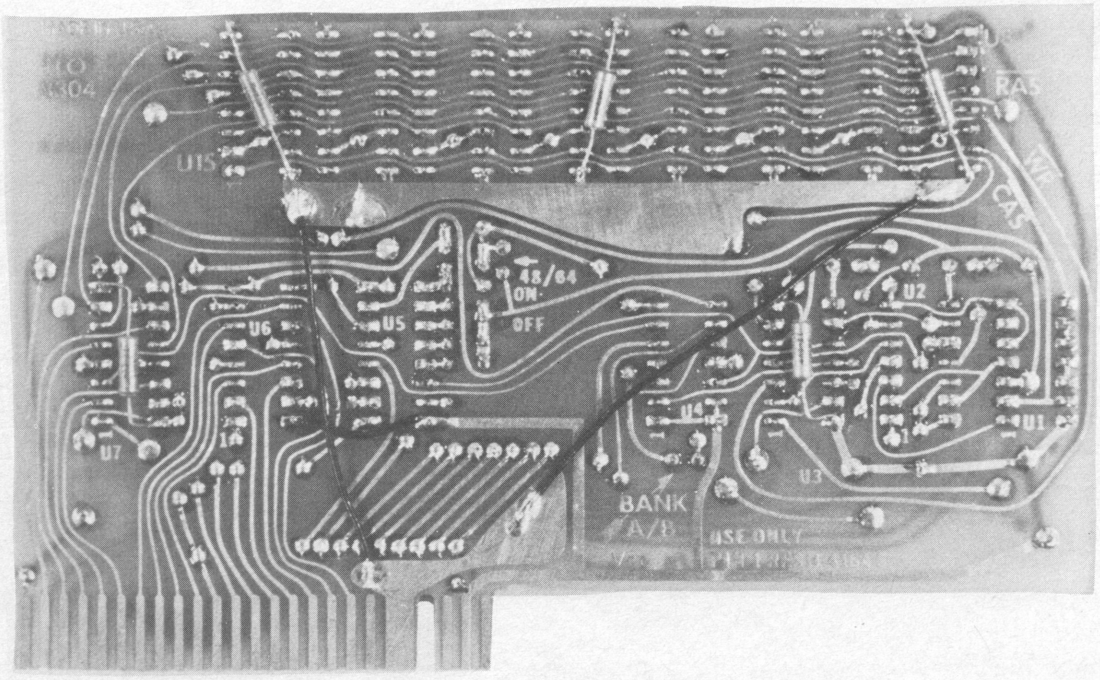
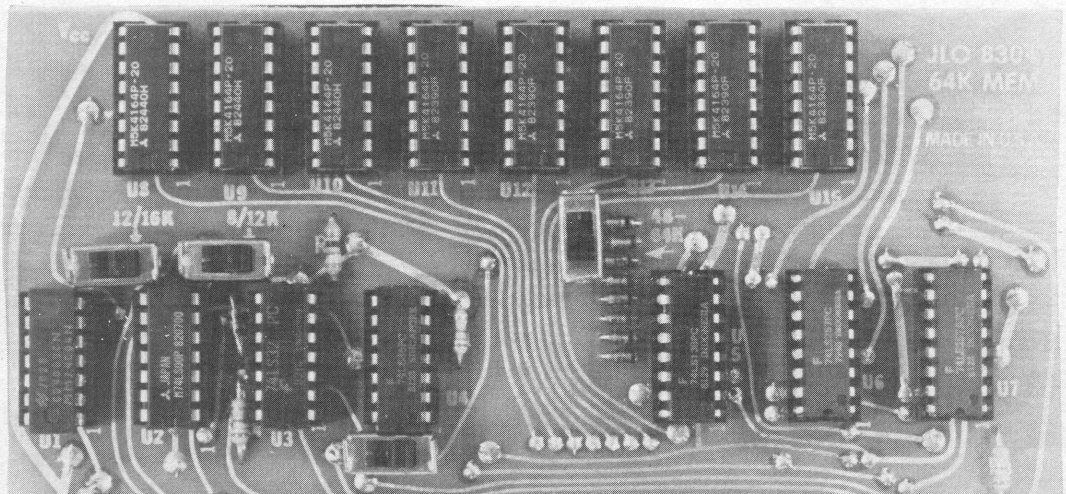


Photo 6.



places with no shorts. Inspect the board for solder bridges or other shorts. Look carefully around the memory array or anywhere that a trace goes between two IC pads. It's easy to get a solder bridge here. Inspect all the feed-throughs to make sure they are all soldered well. Resolder any you are not sure of. One bad joint or short keeps the whole circuit from working.

When your cursor comes onto the screen, enter:

```
PRINT PEEK 16388+PEEK 16389*256
```

If you have not put the ROM on EPROM with the changes in the monitor necessary to check all of RAM on power up, the computer should print 32768. If you have made this change in the ROM, and the 48/68K switch is off, the computer should print 49152. If the 48/64K switch is on with the ROM changes, it should print 65535. If you do not have this change in your system, then enter:

```
POKE 16389,255
POKE 16388,255
```

Make sure the 48/64K switch is on, then enter NEW. When the cursor comes back on the screen enter:

```
PRINT PEEK 16388+PEEK 16389*256
```

You should now see 65535 on your screen. If any of these tests fail, then try 'em again. If they still don't work as they should, check the board for errors. Now enter:

```
POKE 9000,11
POKE 16000,11
```

Move the BANK A/B switch to the other position and enter:

```
POKE 9000,55
POKE 16000,55
```

Now enter the following little program:

```
 10 PRINT AT 10,0;PEEK 9000,PEE
K 16000
 20 GOTO 10
```

RUN the little program and watch the screen as you move the BANK A/B switch back and forth. The computer should print two 11s on the screen when the switch is in one position, and change them to 55s in the other position. If it doesn't, look for board errors.

If the computer does all these tests as it should, GREAT! We are done with the project and it checks out.

Optional Computer Modification

You may like to make one more modification to your TS1000 to use with this board. (SQ did not test this modification.—KO)

As you probably know, the ZX/TS can only use the top 32K of RAM for data storage or for large arrays, because of its video system design. Sinclair only intended this computer to have a maximum of 16K RAM. Also, because of this limitation, we cannot run any machine code (MC) above 7FFFH correctly, as the computer comes from the factory. But, with a very small hardware addition to the ZX/TS, we can run MC from 8000-BFFFH. While this will not work on every 64K memory available, it works on some, including this circuit (yet another reason for building this memory). To do this, you must open the computer up, thus **voiding the warranty**.

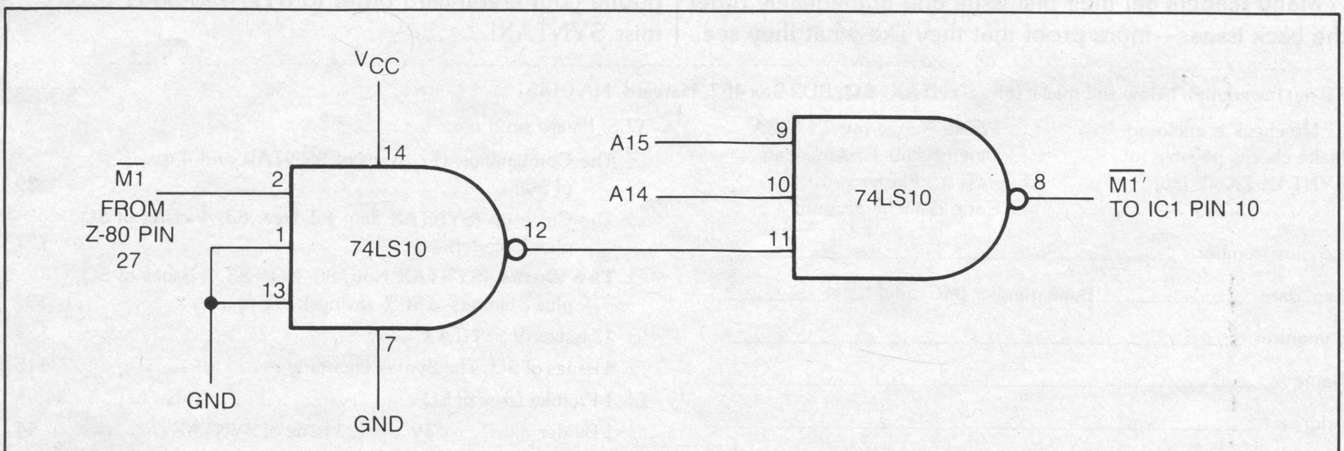
Open the computer case by removing the five screws on its bottom holding it together. Three of these screws are hidden under rubber feet. Now find the Sinclair custom chip (the one that says ULA on it). Carefully remove it from its socket and wrap it in some aluminum foil. Set it aside. This is the computer's most expensive and hardest-to-replace chip. It is also the most easily damaged. Now find the trace going to pin 10 of the ULA. Cut this trace at a convenient place and install the circuit (Schematic 1) on a small universal DIP socket. You can use double sticky foam tape to mount the little board wherever you like. The output of the little circuit (pin 8) connects to the ULA's socket pin 10. Pin 2 of the circuit connects to the trace you cut, the M1 NOT signal from the Z80A pin 27, on the side away from the custom chip. Connect pin 9 of the circuit to pin 18 of the ULA socket and connect pin 10 of the circuit to pin 11 of the ULA socket. You can get Vcc from pin 40 of this socket and ground from pin 34 on the ULA socket.

Look over your connections again to verify that they are correct, check for solder bridges, then carefully reinstall the custom chip. Screw the case back together, and power it up. If the cursor comes on the screen as usual, you made the modification successfully. CONGRATULATIONS. If the cursor does not appear on the screen as usual, open the case back up and find your error. You can again remove the custom chip and then use an Ohmmeter to check continuity of the connections.

You can now run MC from 32-48K. Load in your HOT Z or similar tape and try it.

Theory of Operation

Our last consideration is the memory circuit's theory of operation. We will go through exactly how this circuit operates. It is, of necessity, fairly complex because the circuit is fairly complex. Try to follow it through with the



Schematic 1.

Circuit modification to run machine code from 8000 to BFFFH. Cut trace to pin 10 of custom chip (IC1) near the chip.

SYNTAX

SERVING TIMEX-SINCLAIR PERSONAL COMPUTERS

A PUBLICATION OF **THE HARVARD GROUP**

ISSN 0273-2696

SYNTAX is a monthly newsletter exclusively for ZX80/81/TS1000 owners. We bring you news, reviews and applications for your computer, plus technical notes for circuit-builders. SYNTAX also provides a forum for thousands of users to share advice and problems about programs and vendors. We bring you timely updates about new hardware, software and books. And we cover *all* the Sinclair-technology computers, including the new TS1000.

At SYNTAX we emphasize practicality. You can apply our suggestions even if you aren't sure at first why they work, because we give you complete instructions. Text is clear and easy to understand. SYNTAX readers already know about:

- An automatic phone-dialer they can put together in a few hours
- Syntactic Sums™ to check input for errors
- Programs to explore computer memory
- How to build external additional RAM
- How to add an 8212 I/O chip to control external devices from their computers

And SYNTAX readers like what they get every month. Subscribers know they can depend on us.

After receiving only three issues of SYNTAX, I find that I anxiously await the next. . . keep up the good work!

Martin Irons
Goshen, NY

Congratulations on the brass-tacks, down-to-earth approach of your newsletter. I'll be looking forward to future issues.

Otis Imboden
Washington, DC

Many readers get their first issue and immediately order the back issues—more proof that they like what they see.

You can see what's special about our publication. We work hard to bring you a quality newsletter. We strive to print useful programs of above-average accuracy. As any computer magazine editor can tell you, program listing accuracy is tough to achieve, but we boost our average with every issue. We test each program to make sure it works, it fits in the designated RAM, and it runs when you follow the directions. We print program listings in screen-image format to make it easier for you (it's sure not easier for us!) to enter program accurately. We invented Syntactic Sum™ as an additional aid for you in getting error-free programs. With your subscription you also get access to thousands of other readers, and our staff experts are available by phone to answer your questions or help you solve problems with your machine.

SYNTAX readers get every month:

- Latest news of Z80 hardware and software
- Programs to organize information, calculate, entertain, or instruct
- Do-it-yourself additions
- Clear explanations for beginners

To share the benefits of SYNTAX, just indicate your choices on the order coupon and return it with your choice of payment in U.S. funds. (Please note that additional postage is required for delivery outside North America.)

We are so sure you'll find SYNTAX useful that we promise to refund your entire subscription fee if you aren't satisfied. An unconditional guarantee—you can't lose.

Join the others who stretch the ZX/TS to the utmost. Act now—as soon as we receive your coupon with payment, your first issue will be on its way. For faster service, phone your credit card order to 617/456-3661. Don't miss SYNTAX!

Fill out the coupon below and mail it to: SYNTAX/**SQ,** RD2 Box 457, Harvard, MA 01451 SQ583

☐ My check is enclosed.
Make checks payable to:
SYNTAX ZX80, Inc.

☐ Please charge my ☐ VISA ☐ Diner's Club ☐ American Express ☐ Mastercard ☐ Carte Blanche account

Account number ______________

Exp. date ______ Bank number (MC only) ______

Signature ______________

Name ______________

Address ______________

City______ State ______ Zip ______

Day phone ( ) ______ Evening phone ( ) ______

☐ This is a renewal. My subscription number is: ______________

☐ This is a new subscription.

YES! Please send me:

☐ The Combination (12 issues of SYNTAX and 4 issues of SQ) $39

☐ The Catch-up (SYNTAX Jan. 82-Dec. 83, 4 issues of SQ, plus 1 binder) $77

☐ **The Works** (SYNTAX Nov. 80-Nov. 83, 4 issues of SQ, plus 2 binders, a 50% savings) $97

☐ 12 issues of SYNTAX $29

☐ 4 issues of SQ, The Syntax Quarterly $15

☐ 1 Premier Issue of SQ $4.95

☐ 1 Binder . . . . . $9 ☐ 1 issue of SYNTAX $4

These offers expire 31 Jul 83—SUBSCRIBE NOW.

Increase Your Computer's Utility With Our Help

You own a powerful computer, capable of sorting, analyzing, calculating, displaying and manipulating data as well as measuring and controlling. Despite its small size, your computer will help you learn, analyze your business, keep your records or your schedule, dial your phone, send messages, or talk to other computers. To control the power available to you, you'll need hardware and software to use with your machine. Although you can buy many of these products—and we'll tell you about them—some you must create or modify for yourself. SQ will provide you with the complete information to let you use your programming or tinkering time efficiently.

Our experts also bring you SYNTAX, the newsletter, for up-to-date, concise information. You need news and new product announcements quickly. And SYNTAX packs a lot of information into its brief, time-saving format.

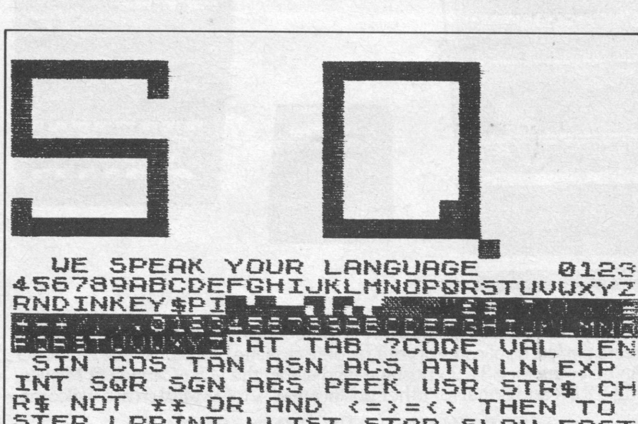
But we recognized your need for material too long to fit in SYNTAX. Syntax Quarterly fulfills that special need. In-depth, extensive, detailed information—that's what SQ's all about. Long programs, big construction articles, and detailed directions help you use your computer to its fullest while learning.

SQ gives you tested, accurate information that you can depend on. We test, we build, we check everything that goes into SQ. Our staff experts work for you; it's as though your full-time staff prepared a report for you every quarter.

At SQ, we specialize in your machine—Timex-Sinclair technology. Our programs and projects work, and, because we test each one before we publish it, they work on *your* machine. All the information in SQ can work for *you*.

But, to capture these benefits—free software, free plans for hardware, free instruction—you must subscribe. Use the coupon to tell us how to help you best. Or call; we understand people in a hurry. **SQ**

```
 100 LET A$="      "
 200 LET B$="      "
 250 LET C$=" "
 300 PRINT A$,A$
 400 PRINT B$,B$
 500 PRINT B$( TO 7),B$
 600 PRINT B$( TO 7),B$
 700 PRINT A$,B$
 800 PRINT " ";B$(2 TO ),B$
 900 PRINT " ";B$(2 TO ),B$
1000 PRINT B$,B$
1100 PRINT A$,A$
1200 PRINT AT 7,22;C$
1300 PRINT AT 9,24;C$
1400 FOR I=0 TO 248
1450 IF I>66 AND I<128 THEN GOTO
 1600
1500 PRINT CHR$ I;
1600 NEXT I
1700 PRINT AT 10,0;" WE SPEAK Y
OUR LANGUAGE "
SYNTACTIC SUM:   22174, 8K ROM
```



617/456-3661

617/456-3661

print of the circuit (Schematic 2) in front of you. Unless you are really into electronics you probably won't understand all of it, but do the best you can. If you do understand it all, great! You could now probably tailor the circuit to your own needs. (You hardware buffs could design yourself a little port circuit to take the place of the BANK A/B switch, hence making this 8K bank of memory available under software control. Experiment. I left that out to keep the circuit as simple as possible, but if you know both the hardware and the software of a computer, the possibilities are endless!)

Let's work our way back from the 4164 memory chip. These devices use eight address lines (thus 256 values from 0-255) whose meaning to the chip depends on the state of four control lines—RFSH NOT, WR NOT, RAS NOT, and CAS NOT. Chips respond by putting data on, collecting it from, or ignoring the data lines. Each of the eight chips stores one data bit of the byte.

Our memory is doubly addressed, if you will. All memory chips receive the address through the multiplexers, but do not interact with the data lines until the chip select line goes low. All chips are simultaneously selected—their CS NOT inputs are tied together. The second addressing occurs because we derive the CS NOT from the address lines as well. Chip select starts the memory cycle.

For memory blocks we don't wish to use, we interrupt the CS NOT signal by opening a switch. Three of the four switches perform this function.

Our fourth switch, BANK A/B select, uses the CS NOT signal for the 8-16K range to alter the apparent address the RAM sees.

Basic Memory Operation

When the CS NOT line goes low, from whatever source, it's inverted by U1 and fed to U2, pins 13 and 2. If the RFSH NOT signal from the CPU is high on U2 pin 12, (we are not doing a refresh) then the output of U2 (pin 11) goes low, strobing the low order addresses from the MA0-MA7 lines into the memory chip with a RAS.

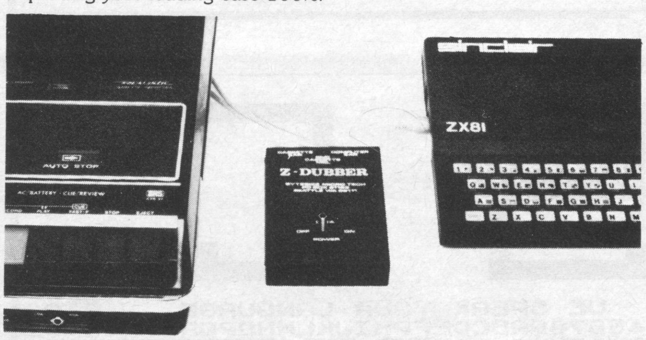
We want to connect the data-in pin (pin 2) to the data-out pin (pin 14) on each chip. To do so, we must prevent conflicts between the read and write control signals. This procedure is known as operating the chips in the "early write state." We control the conflict by assuring that WR NOT goes active (low) at pin 3 of the memory chips before CAS NOT goes active (low) at pin 15.

Since the 4164s use two sets of 8 input pins for addressing, we must multiplex (MUX) the 16 address lines from the computer. We applied A0-A7 as soon as CS NOT went low and we need to apply A8-A15 with a slight delay. U6 and U7 do the switching for us, but we must generate the MUX control.

To get a delayed signal, we apply the inverted CS NOT from U1 pin 4 to pin 2 of U2. There we combine it with a RD OR WR signal from pin 6 of U2 to generate a low at pin 3 of U2. Pins 1 and 2 of U1 invert this signal to make the MUX line high a little after the low order addresses

NO MORE BAD CASSETTE LOADS with THE Z-DUBBER

The frustration of trying to load a cassette program into your Sinclair ZX80/81 or Timex 1000, is this what you really bought your computer for? **Why put up with it? Now you don't have to.** The Z-Dubber is a small device which connects between your computer and cassette player, improving your loading ease 100%.



The Z-Dubber also allows you to connect two tape recorders together, to create perfect duplicates of your favorite cassette programs. The Z-Dubber can be yours for $31.95 postage paid within the U.S., Canada & Mexico. Add $2.50 additional for other postal areas. Visa & Mastercard welcome.

FOR CHARGE AND C.O.D. ORDERS

VISA

1-(800) 227-3800 TOLL FREE
1-(800) 792-0990 IN CALIFORNIA
ASK FOR OPERATOR 225

master charge

For additional information or service, call or write:

BYTESIZE MICRO TECHNOLOGY
PO BOX 21123 - DEPT. B - SEATTLE, WA 98111
(206) 236-BYTE

Dealer Inquiries Wanted.

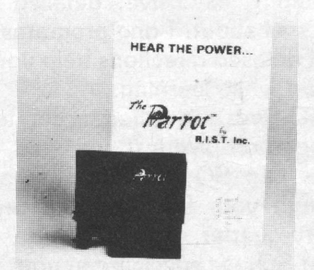
R.I.S.T. INC. **ANNOUNCES** FREE SUBSCRIPTION TO SQ

PURCHASE A PARROT SPEECH SYNTHESIZER . . .
YOU'LL RECEIVE:
SUBSCRIPTION TO SQ **"$15 VALUE FREE"**
EXCLUSIVE PHRASE FINDER PROGRAM **"$5 VALUE FREE"**
TALKING CASINO GAME CARTRIDGE **"$15 VALUE FREE"**

NOW YOUR TIMEX SINCLAIR CAN TALK.
QUICKLY. SIMPLY. ECONOMICALLY.

AT LAST A *QUALITY SYNTHESIZER.*
PROFESSIONAL JOURNALISM.
UNBEATABLE BARGAIN!

The Parrot™



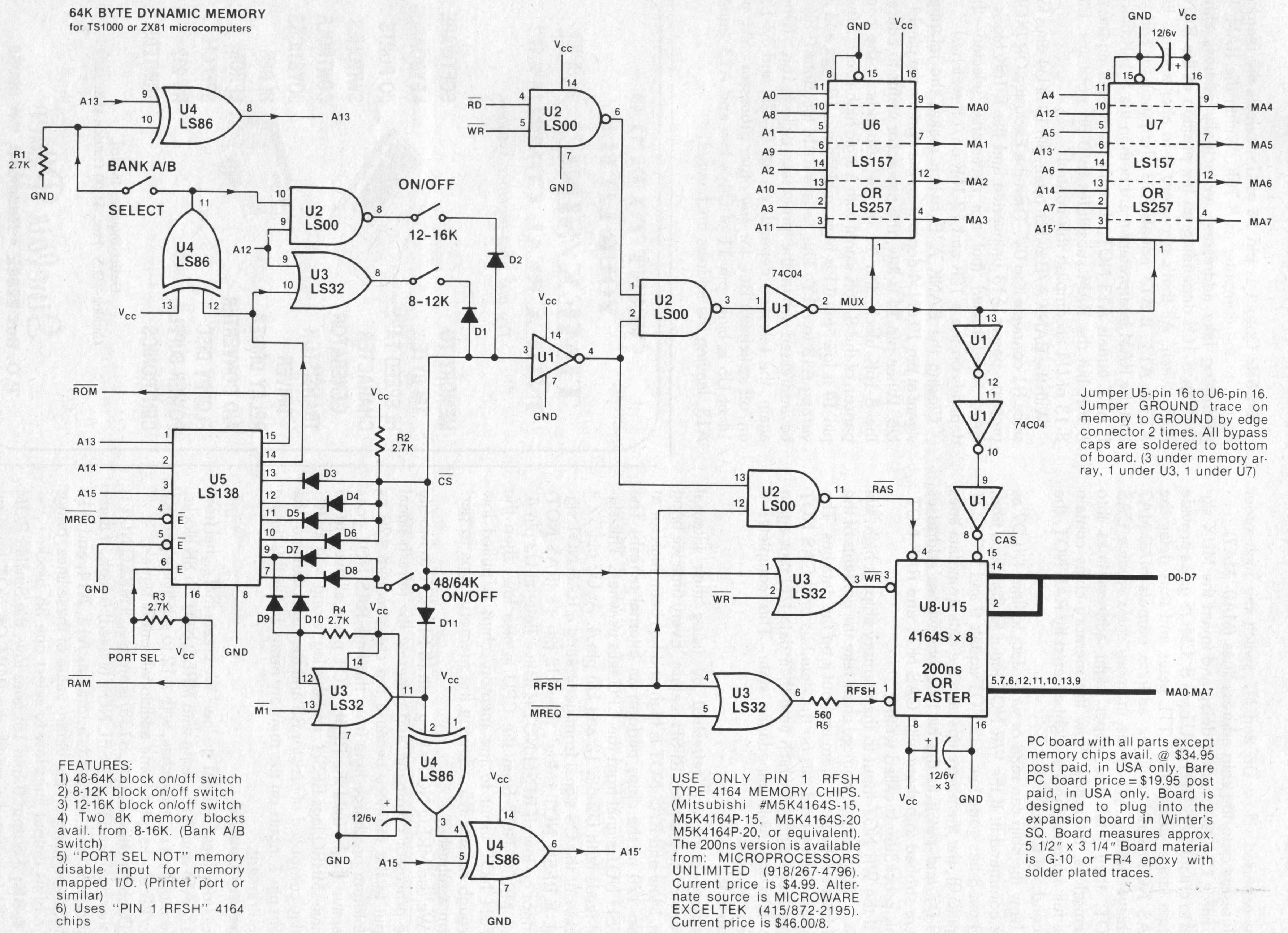
- **PROGRAMMABLE SPEECH AND SOUND EFFECTS.**
- **AMPLIFIER AND VOLUME CONTROL.**
- **"PIGGY BACK" EXPANDABILITY.**
- **SPEAKER OUTPUT JACK.**
- **POWER INPUT JACK FOR EXPANDED SYSTEM OPERATION.**
- **COMPLETE DETAILED INSTRUCTIONS.**
- **ALL FOR $89.95.**

LET THE POWER OF YOUR SINCLAIR ZX80/81 TM OR TIMEX SINCLAIR 1000 TM SPEAK FOR YOU NOW. THROUGH R.I.S.T. COMPUTER COMPONENTS, INC. THE PARROT TM – YOUR ZX80/81 OR TS1000 CAN SPEAK . . . ***CLEARLY, CONSISELY AND WITHOUT LIMIT.***

R.I.S.T. HAS DEVELOPED A STATE OF THE ART VOICE SYNTHESIS MODULE ***YOU PLUG IN.*** *THE PARROT TM UTILIZES ALLOPHONE CONCATENATION TO SYNTHESIZE SPEECH, WITH LIMITLESS VOCABULARY, SENTENCES, AND ANY ENGLISH LANGUAGE WORD CAN BE CREATED.*

BEST OF ALL, THE PARROT TM IS AFFORDABLE, FOR $89.95, YOUR ZX80/81 OR TS1000 CAN GIVE YOU THE VOICE OF THE FUTURE . . . ***TODAY!***

SEND $89.95 PLUS $4.00 SH/HD TO:
RIST INC.•DEPT. 4-5•P.O. BOX 499•FT. HAMILTON STATION, BROOKLYN, N.Y. 11209
N.Y. STATE RESIDENTS ADD 8.25% TAX ($7.40).
MY COMPUTER IS: ☐ ZX80 ☐ ZX81 ☐ TS1000
☐ 9V,2 Amp POWER SUPPLY (OPTIONAL) $24.95 ea.
(REQUIRED FOR ZX80 OR EXPANDED SYSTEM OPERATION)
FOR ORDERS OUTSIDE U.S. PLEASE SEND CHECK PAYABLE IN U.S. DOLLARS


64K BYTE DYNAMIC MEMORY
for TS1000 or ZX81 microcomputers
BANK A/B
SELECT
ON/OFF
12-16K
8-12K
48/64K ON/OFF
U4 LS86
U2 LS00
U3 LS32
U1 74C04
U5 LS138
U6 LS157 OR LS257
U7 LS157 OR LS257
U8-U15 4164S × 8 200ns OR FASTER
MUX
D0-D7
MA0-MA7
Jumper U5-pin 16 to U6-pin 16. Jumper GROUND trace on memory to GROUND by edge connector 2 times. All bypass caps are soldered to bottom of board. (3 under memory array, 1 under U3, 1 under U7)
PC board with all parts except memory chips avail. @ $34.95 post paid, in USA only. Bare PC board price = $19.95 post paid, in USA only. Board is designed to plug into the expansion board in Winter's SQ. Board measures approx. 5 1/2" x 3 1/4" Board material is G-10 or FR-4 epoxy with solder plated traces.
USE ONLY PIN 1 RFSH TYPE 4164 MEMORY CHIPS. (Mitsubishi #M5K4164S-15, M5K4164P-15, M5K4164S-20 M5K4164P-20, or equivalent). The 200ns version is available from: MICROPROCESSORS UNLIMITED (918/267-4796). Current price is $4.99. Alternate source is MICROWARE EXCELTEK (415/872-2195). Current price is $46.00/8.
FEATURES:
1) 48-64K block on/off switch
2) 8-12K block on/off switch
3) 12-16K block on/off switch
4) Two 8K memory blocks avail. from 8-16K. (Bank A/B switch)
5) "PORT SEL NOT" memory disable input for memory mapped I/O. (Printer port or similar)
6) Uses "PIN 1 RFSH" 4164 chips

Schematic 2.

are captured. Now U6 and U7 switch the high order addresses onto the chip address lines (MA0- MA7).

Three CMOS inverters delay and invert the MUX signal to create CAS NOT at U1 pin 8. CMOS gates provide much longer delays than TTL gates and thus delay the CAS NOT signal until the address outputs MA0-MA7 achieve stability. When the memory chips receive CAS NOT, they latch the upper eight address lines into themselves, internally. Now the memory chips contain the full address, the first eight entered by RAS NOT, the second, by CAS NOT.

Now the chip can read or write data to lines D0-D7 as we command it. If the WR NOT input to the chips from U3 pin 3 has gone low (CS NOT equals 0 and WR NOT equals 0), the memory chips take the data from data lines D0-D7 and put it in the bit locations whose addresses were strobed into them with CAS NOT and RAS NOT. This procedure is called writing.

If the WR NOT input to the memory chips is high, we are reading, so the memory chips take the contents of the address strobed into them and put it on the data bus. The memory chips hold this on the data bus until CAS NOT again goes high, at which time their data I/O pins again go into the high impedance state. This completes a memory cycle.

Refresh

Memory chips are refreshed by using their internal counters via the pin 1 RFSH feature. Every time we bring pin 1 low and RAS NOT is high and has been high for at least 120 ns the chips perform an internal refresh. The RFSH NOT signal from the CPU tells us when. This signal feeds both U2 pin 12 and U3 pin 4. At U2 pin 12 it locks out the CS signal from generating a RAS NOT signal. If RFSH NOT is low, then U2 pin 11 (RAS NOT) cannot go low. If RFSH NOT is active (low) at U3 pin 4, and MREQ NOT from the CPU is also low, then the RFSH NOT input to the memory chips is pulled low through resistor R5, allowing the memory chips to perform a refresh.

We use MREQ NOT along with RFSH NOT to modify the processor command and generate the refresh signal input to the memory because the chips require 120 ns minimum precharge time from the time RAS NOT goes inactive (high) until the RFSH NOT pin is allowed to go active. MREQ lags RFSH, and this lag, along with an RC delay composed of resistor R5 and the capacitance of the eight pin 1 inputs on the memory array, fulfills this timing requirement.

Memory Control

Just how does CS NOT go low? We create that signal from A13, A14 and A15, along with MREQ NOT, in 8K blocks. We modify it with switches, with A12 to subdivide part of it into 4K blocks, plus some additional control signals.

Decoder chip U5 (74LS138) is enabled if MREQ NOT is active (low) and PORT SEL NOT (if used) is high. If these conditions exist, U5 reads addresses A13, A14, and A15 and, depending on the binary value of these lines, brings one of its output lines low to select one 8K block.

0-8K: If output 0 (pin 15) goes low, the Sinclair ROM (0-8K) is selected through the ROM CS NOT trace on the edge connector.

8-16K: Output 1, pin 14 of U5, selects the memory known to the computer as 8-16K. Within the RAM, by generating fake addresses, we actually address either 0-8K or 8-16K depending on the position of the BANK A/B switch. All that's necessary is to alter address line A13 to A13'. If A13' equals A13, the processor address and the RAM interpretation are both in the 8-16K block. If A13' equals A13 NOT, then the processor addresses 8-16K but the RAM substitutes its 0-8K positions. Pins 8-13 of U4 accomplish this control.

With the BANK A/B switch open, pin 10 of U4 stays low since R1 connects it to 0V. Then the Exclusive OR (XOR) property passes A13 uninverted and the 8-16K block of RAM is addressed (but not necessarily selected—that depends on the 8-12K and 12-16K on/off switches).

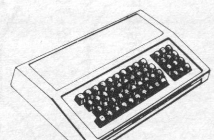
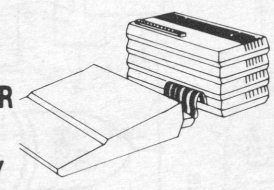
Closing the BANK A/B switch makes the reference signal at pin 10 of U4 depend on the signal from pin 14 of U5. When pin 14 is low, the processor wants to address the 8-16K block. Pin 11 of U4 responds to the comparison of this 8-16K signal with a logic high (Vcc). When pin 14 is low, pin 11 is high, making pin 10 high and inverting A13 to A13'. This causes RAM locations 0-8K to be addressed when the processor calls 8-16K. (Once again, the selection depends on the 8-12K and 12-16K on/off switches.) But, for any other processor calls, pin 14 of U5 is high, pin 11 of U4 is low and A13' equals A13, leaving the address unchanged.

WANT TO BUILD YOUR LITTLE

TIMEX/SINCLAIR

INTO A REAL COMPUTER?

Try our plug-together systems.

MEMORY TO 1M BYTE
HI SPEED TAPE
CHARACTER GENERATOR
TRANSISTOR DRIVER
RELAY DRIVER
A/D CONVERTER
FLOPPY DISC
POWER SUPPLY
CENTRONICS

SOFTWARE
KEYBOARDS
I/O PORTS
SWITCHES
CONTROLS
JOYSTICKS
HI RES
EPROM
SPEECH
RS 232
PRINTERS

Dealer Inquiries Invited.
Catalog—U.S. Free, $1.00 Outside U.S.

Sinclair Place

P.O. Box 2288E • Redmond, WA 98052

Address line A12 controls the choice of 8-12K or 12-16K, using one gate of each of U2 and U3. If the switches for these blocks are open, no memory is selected for these addresses, but the memories are still refreshed. Thus you can write information here and then open the switches without losing the stored data.

When the output of U5 is low, pin 11 of U4 is high. Thus, if A12 is low, U3 responds by making its pin 8 low to provide a potential CS NOT through the 8-12K switch and D1. If A12 is high, U2 responds by making its pin 8 low to provide a source of CS NOT through the 12-16K switch and D2. In all other cases, both pins 8 are high. As previously described, the RAM addresses that would respond lie between 8-12K if the BANK A/B switch is open and between 0-8K for a closed switch.

16-48K: If outputs 2, 3, 4 or 5 (pins 13, 12, 11 and 10) of U5 go low, the CS line is pulled low directly, through diodes D3, D4, D5 and D6. One of these lines will go low if we are accessing the 16K-48K memory block. Other than pulling the PORT SEL NOT input on the U5 low, there is no way to disable this memory block.

48-64K: Outputs 6 and 7 control the 48-64K range unless M1 NOT is low. If M1 NOT is low, we generate an output from the display file in the 16-32K range.

Suppose the 48-64K switch is closed and M1 NOT is high. Then output 6 or 7 pulls CS NOT low through D7 or D8. We get a normal access to the highest memory block. If the switch is open and M1 NOT is high, no memory responds to these addresses.

When M1 NOT is low, the switch position does not matter. An alternate path controls CS NOT and at the same time diverts the address to the 16-32K block whenever 6 or 7 go low.

Outputs 6 and 7 control input pin 12 of U3. If either 48-56K or 56-64K is requested (6 or 7 low), then U3 pin 11 goes low, pulling CS NOT down through D11. This action also controls the conversion of A15 to A15' between pins 5 and 6 of U4. Pin 4 of U4 gets an inverted signal from pin 11 of U3 (created between pins 2 and 3 of U4). When U4 pin 4 is high (M1 NOT low), A15' equals A15 NOT, and the RAM responds from the display file in the 16-32K block. For all other cases, A15' equals A15.

Parts List for the 64K Memory Board

U1 74C04 Hex CMOS Inverter
U2 74LS00 Quad 2-Input NAND
U3 74LS32 Quad OR Gate
U4 74LS86 Quad XOR Gate
U5 74LS138 1 of 8 Decoder
U6 74LS157 or 74LS257 Quad 2 to 1 Decoder
U7 74LS157 or 74LS257 Quad 2 to 1 Decoder
U8-U15 Pin 1 RFSH Type 4164 Memory chips. 200 ns or faster Mitsubishi M5K4164S-20, M5K4164S-15, M5K4164P-20, M5K4164P-15, Mostek MK4164, or Motorola MCM6664.
R1 2.7K, 1/4 Watt resistor
R2 2.7K, 1/4 Watt resistor
R3 2.7K, 1/4 Watt resistor
R4 2.7K, 1/4 Watt resistor
R5 560 Ohm, 1/4 Watt resistor
D1-D11 1N914 or 1N4148 diode
SW1-SW4 SPST Subminiature slide-type switches with one end lead cut off flush
C1-C5 12uF, 6V Axial lead Tantalum capacitors

Miscellaneous

Four 14-pin soldertail IC sockets
Eleven 16-pin soldertail IC sockets
2 feet 30AWG wire for feed-throughs
9 inches 22AWG wire for jumpers

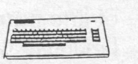
The following is available from:
John Oliger
10115 Nassau Lane
Indianapolis, IN 46229

Complete kit of all parts listed above except memory chips. Includes PC board: $34.95 postpaid in USA. Bare board only: $19.95 postpaid in USA. Indiana residents must add 5 percent state sales tax. Board is G-10 or FR-4 epoxy with solder-plated traces. Two-sided board does not have plated-through holes. Prices include shipping by first class mail.

Tools Necessary for Board Assembly

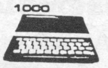
25 Watt or comparable low power soldering iron with a fine tip.
Pair of Radio Shack "Nippy cutters" or similar. Some very small diameter (18 gauge or less) rosin core solder.
A pair of small round-nose pliers (for bending bypass capacitor leads as necessary).
Pair of small wire strippers.
Acetone for flux removal. **SQ**

GENERAL SYSTEMS CONSULTING
2312 Rolling Rock Drive
Conley, Georgia 30027

VIC 20


CASSETTE SOFTWARE

SINCLAIR ZX81
TIMEX SINCLAIR 1000
COMMODORE VIC20

TIMEX SINCLAIR 1000


DESIGNED TO HELP MONITOR YOUR FINANCES
16K MINIMUM FOR T/S 1000 & ZX81

* At least 3K expansion ** At least 8K expansion	T/S 1000 ZX81	VIC 20	ENTER PRICE
AMORTIZATIONS	12.95	13.95	
BAR CHARTS	13.95	14.95**	
ANNUITY EVALUATION	12.95	12.95	
FILE MANAGER	12.95	14.95*	
BANK STATEMENT BALANCER	12.95	13.95	
CHECKBOOK SIMULATOR	12.95	NA	
DEPRECIATION STRAIGHT LINE	12.95	13.95	
DEPRECIATION DECLINING BALANCE	13.95	14.95	
DEPRECIATION ACRS	14.95	15.95*	
DIET PLAN	11.95	NA	
HOME BUDGET	13.95	14.95**	
HOME INVENTORY	12.95	14.95	
HOME PAYABLES	12.95	NA	
HOME EQUITY EVALUATION	12.95	13.95	
REAL ESTATE INVESTING	13.95	14.95**	
SAVINGS INVESTMENT ANALYSIS	13.95	14.95**	
IRS 1040 LONG FORM	15.95	16.95**	
IRS 1040A SHORT FORM & 1040 EZ	14.95	15.95**	
INCOME TAX PROJECTIONS	14.95	15.95**	
IRA ANALYSIS	NA	12.95	

NAME ______	TOTAL YOUR PRICE	
ADDRESS ______	POSTAGE/HANDLING	1.50
CITY/STATE ______	TOTAL PRICE	

SQ SOFTWARE REVIEW

The ZXADream

Name: ZXAD Assembler and Debugger
Type: Utility
ROM/RAM reqd: 8K/16K
Printed listings? No
Program listable? Yes
Easy to load? OK
Written in: BASIC and machine code
Display: Good
From: Scientific Software
6 W. 61 Terrace
Kansas City, MO 64113
Price: $14.95

ZXAD, Scientific Software's assembler/debugger for the ZX/TS, is primitive as assemblers and debuggers go. But compared to hand assembling machine code programs and entering them with a loader or monitor, ZXAD is a dream; it pays for itself in avoided aggravation after one program.

Three utilities comprise ZXAD: an assembler, a debugger (really a monitor that lets you set one breakpoint) and a line-renumbering routine. You select each by typing a single letter in response to the prompt that appears after you RUN ZXAD.

Assembler

ZXAD's two-pass assembler is simple but adequate. On its first pass it builds a symbol table. On its second, it assembles the machine code, or *object code*. ZXAD allows the use of labels and symbols (making it unnecessary to calculate displacements and jumps; you can DJNZ LOOP rather than DJNZ $E0), comments, and the following pseudo-ops:

ORG	establishes location at which to start loading machine code.
EQU	sets up symbol values.
DEFB	gives hex values to one or more bytes (8 bits).
DEFW	gives hex values to one or more words (16 bits).
DEFS	sets aside storage.
DEFC	gives string values up to 256 bytes.
END	signals end of assembly.

Debugger

Essentially a machine code monitor, the debugger lets you look at or modify the contents of any address. It displays the hex value and the character that would be printed. It also allows you to examine the register in a limited way—by inserting a breakpoint, which displays the registers contents at breakpoint when you execute the program. You cannot use the debugger to step through a program as it executes, nor can you insert more than one breakpoint at a time. The debugger does not prevent runaway programs—if your program goes into an infinite loop, you still have to pull the plug.

Line Renumbering

ZXAD's line-renumbering routine helps you enter your Z80 instructions, or your *source code*. These instructions are stored as normal text in REM statements, one instruction to a line, using Z80 mnemonics. You enter numeric constants in hexadecimal, with a dollar sign ($) preceding hex digits. With the line-renumbering routine you can change any section of line numbers. This gives you more flexibility when entering source code, but can result in strangely numbered programs if you aren't careful.

ZXAD assembles the entire Z80 instruction set. Depending on the option you choose, after it assembles the code it prints a listing on the screen, or displays only the error messages and error count. If you assemble the code into a REM statement immediately after ZXAD, your first character resides at address 23330 ($5B22). You can relocate a program by changing the ORG statement and reassembling, and you can insert or delete instructions without having to recalculate jumps or calls—the assembler does it for you. You cannot enter numbers in decimal, nor use expressions in symbols, features which would have been useful. ZXAD has no macro capability.

Using ZXAD

ZXAD comes with a 17-page booklet that describes its operations and lists the Z80 instructions and ways you can enter them into your source code. The documentation seems adequate, though one complete example program showing all the assembler's features in use would have helped.

Follow the booklet's warnings, especially those pertaining to confusing 8- and 16-bit symbols and numbers; the syntax checker does not check for this error, though it does let you know if a relative jump is too far. You will probably get bad values in your object code if you try to enter an 8-bit number where you should enter a 16-bit number and vice versa.

Some of ZXAD's limitations are easy to overcome. I added a hex-to-decimal and decimal-to-hex routine, plus a routine to let me execute a program by giving its hex address rather than having to convert to a decimal address. It was not difficult to incorporate these features into the menu. I plan to also add a routine that enables me to write the object code out to a file. Using an unmodified ZXAD to put just the machine code program on tape, you must write the object code to a REM statement, patch the object code to change any absolute addresses, delete the source code and the assembler, and then save what's left.

Despite its limitations, ZXAD appears to be a reliable assembler and it serves its purpose well. It eliminates most of the aggravations and frustration encountered in machine code programming. For $14.95 it's a bargain and I highly recommend it.

David Bookbinder
Cambridge, MA

The Miracle Worker

Name: ZX Compiler--Integer BASIC Compiler
ROM/RAM reqd? 8K/16-64K
Price: $22 tape and manual
From: Bob Berch
19 Jaques St.
Rochester, NY 14620

Machine code (MC) is the key to compact, fast-running programs. Yet, to write entire programs in machine code takes a thorough understanding of Z80 assembly language. Even if you can write short routines, an overlying BASIC program usually runs the show. If you want to write MC programs you may find the solution in ZX Compiler.

Simply put, Bob Berch's little miracle worker converts BASIC into machine code. You can compile short programs and use them as USR calls in another BASIC program, or you can compile entire programs.

Yet, ZX Compiler can't do it all. For example, you can only use strings in PRINT, and string arrays are not allowed. You can use up to 255 variable names—but only one or two characters long.

All arithmetic is integer and you can't use floating point functions. But, you can use trig functions such as LN and INT.

System commands not permitted include: LLIST, NEW, CONT, LOAD, LIST, RUN, SAVE and CLEAR. However, you can use the keyboard printer commands, so ZX Compiler works with any ZX/TS-compatible printer.

OK, so you have to think up new ways to do a few things. But it's not all bad. ZX Compiler adds two new features to BASIC. For example, MOD function puts limits on RND and lets you input and print in either decimal or hexadecimal.

ZX Compiler comes in two versions on one cassette: a hi-memory version that starts at 28672 for 16K RAM, and a low-memory version that resides at 12288 if you can access the 12 to 16K block of memory.

ZX Compiler's menu-driven program offers five routines: initializer, compiler, 1 REM generator, code mover, and run-time relocator.

Each time you initialize, the program requests answers to four questions. First, you decide if you want to communicate in decimal or hexadecimal. Then you choose where you want to store the compiled code (default is a 1 REM) and set the location of the routine package. Finally you pick the variable storage area. However, ZX Compiler can make three choices for you automatically. You must choose whether you want decimal or hex. The program, unless otherwise instructed, puts the compiled code at 16514 in a 1 REM followed immediately by the variables and the run-time package.

After you choose the parameters, the compiler converts programs written in BASIC to machine code. It *destroys* your BASIC program in the process so make an extra copy before compiling.

If the compilation is not successful, the compiler generates an error report you can use to find the problem. The ten error reports, numbers 0 through 8 and 10, specifically identify your problem.

REM generator creates a REM of any desired length, subject to available memory.

ZX Compiler's code mover routine lets you transfer code intact from one section of memory to another. It cannot change internal addresses.

Once your program is compiled it needs the support of the run-time package. This 768-byte machine code routine consists of subroutines used by calls from the compiled code. The run-time relocator lets you move the run-time package in memory and relocates absolute addresses within itself so it can operate wherever you put it. Only subroutines that your program needs are required in the run-time package. For example if your program does not need the INPUT function, you can eliminate that subroutine from the run-time package and save almost 200 bytes. An appendix in the manual lists all 39 routines' starting addresses and explains how to delete them.

For my first try with ZX Compiler I decided to write a utility to convert numbers from decimal to hexadecimal and back. The task proved even simpler than I anticipated because the conversion routines are a part of the run-time package.

After writing and saving my BASIC program, I called up the initializer. I selected the decimal mode and entered 8260 as the final origin for the compiled code. (I run a Memopak 64K, which reclaims the 8-16K block for MC.) Next, I specified the final location of the run-time package, 8192 (just ahead of the compiled code) and set the variables storage following the compiled code.

I entered a call to the compiler and, in an instant, it did the job. Before moving the code and run-time package to their final locations starting at 8192, I tested the results of my labors. Voila, the program ran without a hitch!

Using the REM generator I created a REM large enough to hold the run-time package plus compiled code and variables, then filled it using the code mover. I added a short routine to move the compiled code, variables and run-time package to low-memory when called. The job was done. Now, I just load my conversion program and it relocates itself into low-memory, ready to use.

I am pleased with ZX Compiler. Its shortcomings are offset by the increased capabilities it gives me, an average programmer.

I would like to see more space in the 19-page manual devoted to overcoming problems of converting BASIC programs to compilable form. Other than that, the manual is very complete. It includes addresses of all subroutines in the run-time package and gives 23 examples of what BASIC code compiles into.

ZX Compiler is not for rank beginners to machine code, but you don't need a lot of sophistication to benefit from it. For users unfamiliar with or not proficient in machine code, this program is a valuable tool.

Maynard M. Kealiher
Aurora, CO

Gammon Your ZX/TS

Name: Backgammon
Type: Game
ROM/RAM reqd: 8K/16K
From: Sinclair Research Ltd.
3 Sinclair Plaza
Nashua, NH 03061
Written by: Psion Software Ltd.
Price: $9.95

To be entirely fair, let me say that no good commercially available backgammon game exists for any computer system, at least not one that plays as well as the average chess program plays chess.

Backgammon is difficult to program, especially in limited memory. Still, I investigate each new program, hoping for a surprise.

I was not surprised with ZX81 Backgammon from Psion.

Perhaps for beginners this program may have some value. But for anyone who has played the game a dozen or more times, it is worthless.

A main fault with this and many other backgammon programs is the safe playing style early in the game. The system plays a game of avoiding blots at all cost and loses flexibility. Aggressive play beats this style almost every time. After ten games I was beating it 42 to 3 at the highest level of play. It loses gammons frequently.

ZX81 Backgammon uses the doubling cube. It uses it legally, but feeble-mindedly. On two occasions it doubled me when I had a closed board and it had a man on the bar!

On the plus side, the display is nice, and entering moves is easy enough. The tape loads with no problems.

But it is slow. It seems to take forever to show your move on the display after you enter it. Then you must wait again for the computer to move. This is in FAST mode.

I have tried to program backgammon myself, unsuccessfully. I know it is far more difficult to program than chess. But strong programs have been written for larger systems. When will a good one be available for the home?

Gary Wolf
Clifton, NJ

Big Graphics—Small Challenge

Name: Mega Mind
Type: Game
ROM/RAM reqd: 8K/16K
Printed listing? No
Program listable? Yes
Written in: BASIC and machine code
Challenge: Average
Display: Excellent
From: Orbyte Software
P.O. Box 948
Waterbury, CT 06720
203/753-8308
Price: $19.95

This program is a spectacular tour-de-force of graphics. If you want a graphic demo program or want to study how Orbyte accomplished this with machine code, then consider buying this program.

Mega Mind takes 6 minutes and 15 seconds to load and occupies over 15K of memory. It runs automatically after loading and opens with an excellent graphic display.

When you actually get down to playing the game you discover that it's nothing more than the old code-breaker game and is limited to four digits out of six numbers at that. The only new variation is that you and the computer play simultaneously, trying to guess each other's code first. As added and unnecessary complication, you enter your guesses as numbers, but they appear on the screen as fancy symbols. You may get confused, but of course the computer never does.

If you are looking for a mental challenge, I'm afraid this game is not it. My disappointment resulted mainly from my high expectations because of its price and its advertising copy. If Mega Mind's designers applied their obvious graphic talents to making a more challenging game, they could have a real winner.

Peter D. Hoffman
Corpus Christi, TX

2-BIT™ SOFTWARE, P.O. Box 2036
Dept. SQ1, Del Mar, CA 92014
(619) 481-3629
DEALER INQUIRIES INVITED

LEARN TO PROGRAM

Text and File Organizers with ZX DATA FINDER

A high capacity information storage and retrieval tool for 16 K Timex and Sinclair Computers.

Advanced file input and editing routines are thoroughly analyzed.

Comprehensive search and display methods are fully explained.

AN ADVANCED COURSE IN DATA HANDLING

Free specifications are available, or send $9.95 for program listing and text to:

THOMAS B. WOODS
P.O. Box 64, Jefferson, N.H. 03583

CHANGE YOUR
TIMEX/SINCLAIR 1000 DISPLAY
TO FULL <u>COLOUR</u> GRAPHICS
with

KOLORWORKS

LOOK AT THE FEATURES ! ! !

- Plugs into ZX81/1000 (edge connector)
- Latest technology with TMS9918 VDP (32 sprite levels)
- Module contains it's own memory
- All text will run on the color tv
- User defined characters & graphics up to 256x192 pixels
- Module contains extension of basic commands including: PAPER/INK/BORDER/BIN/SPRITE/OUT/INP/etc.

for $149.95

KOLORWORKS COMES WITH A LIMITED WARRANTY ON PARTS AND WORKMANSHIP

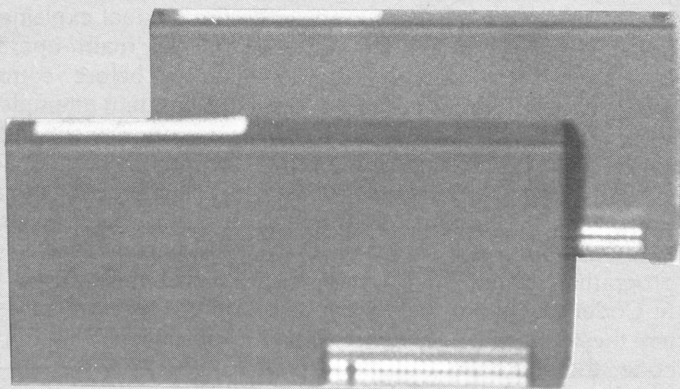
USE YOUR KOLORWORKS IMMEDIATELY WITH A GAME CASSETTE FOR $9.95

This delightful game is designed for hours of fun using some of the color graphic capabilities of KOLORWORKS. The cassette also contains a short program to familiarize you with some of the commands and graphics.

Enjoy Game Fun With

GAAMWORKS

THE SOON TO BE RELEASED GAME MODULE (proto-type stage) WILL OFFER SOUND, ROM CARTRIDGES AND JOY STICK PORTS FOR YOUR TS1000/ZX81.



- **THE SOUND** will be of arcade game quality which you can program for music, animals, transportation (auto, train, airplane, etc.) and machine sounds.
- **ROM CARTRIDGES** will have up to 8K of ROM using either 2716, 2732 or 2764 EPROMS. We will have pre-programed cartridges and blank cartridges which you can program. We will be offering a service to burn EPROMS from your cassettes.
- **JOY STICK PORTS** will allow for the use of two "Atari"®compatible joy sticks.

SORRY PRICE IS NOT AVAILABLE AT THIS TIME! GAAMWORKS WILL BE AVAILABLE BY MAIL ORDER FOR FURTHER INFORMATION, SEND $2.00 (Credited to Order).

At this time KOLORWORKS and GAAMWORKS is available only by mail order.

MAIL TO: **BRAINCHILD COMPUTER WORKS, INC.**
P.O. Box 506
Pewaukee, WI 53072

	PRICE	QTY.	AMOUNT
KOLORWORKS	$149.95		
CASSETTE	9.95		
Shipping & Handling	4.95		4.95
Wi. residents add sales tax	TAX		
	TOTAL		
My ☐check ☐ money order is enclosed	ENCLOSED		

Name ____________________

Street ____________________

City ____________ State ________ Zip ________

Please allow six to eight weeks for processing. Thank you.

M(inus) Coder

Name: M-Coder
Type: Machine language compiler
ROM/RAM reqd: 8K/16K
Written by: Personal Software Services
482 Stoney Stanton Rd.
Coventry CV6 5FE
England
Price: $16.00

When I read the advertisement for M-Coder, I thought it was too good to be true. Here was a machine code compiler for my ZX81 which, according to the ad, would permit me to "write (my) program in BASIC (or load an existing program), press a key and...automatically compile it into machine code." Further, the ad claimed that "M-Coder handles 99.9 percent of BASIC."

I wasted no time in sending in my order. Although relatively proficient in BASIC programming, I know very little about machine code programming except that it requires less memory and allows programs to run faster than BASIC. Many of my BASIC programs contain subroutines that demand significant amounts of memory, retarding the programs' speed. I thought that M-Coder would be ideal for converting these subroutines into machine code, thereby enhancing these programs' performance.

My M-Coder package arrived in about seven weeks—not bad for an overseas order. I received a cassette tape of the M-Coder program and a manual with four sparse pages of documentation. After reviewing the manual I began to believe my initial reaction to M-Coder's ad had been correct. The manual lists the commands that control M-Coder and provides a brief (and in some cases inadequate) explanation of how to use the commands. For three commands, ABS, USR and INT, it provided no explanation. Four short paragraphs explain how to compile a BASIC program using M-Coder. The manual offers no examples and does not discuss how to use a compiled program as a subroutine.

In the manual you find that M-Coder "runs only integer BASIC being restricted to numbers in the range −32768 to 32767 with no strings or boolean operations." Further examination revealed that "only one array is available in M-Coder" and that "the arithmetic permitted is restricted to the four standard operations (/, *, +, −)." The manual also warns that "you will probably have to make some changes to your (BASIC) program before it will compile properly," but fails to identify these changes.

My M-Coder program loaded from tape on the first try. After several frustrating minutes I was able to compile a three-line BASIC program that assigned values to two numeric variables and summed them. The process would have run much faster had the manual explained that you must put all math operations in parentheses before compiling. All my subsequent attempts to use M-Coder to compile subroutines in BASIC programs proved fruitless, because each subroutine violated at least one and in most cases several M-Coder restrictions. Had I succeeded I still would have faced the problem of how to integrate the compiled subroutines within a main program.

In summary, M-Coder's restrictions and poor documentation will severely limit my use of this utility.

John B. Carson
Takoma Park, MD

Clem Wehner of Scott AFB, IL, details M-Coder's limitations:

It *cannot* handle such important and common operations as:

- More than one string variable
- Alphanumeric arrays
- More than one numeric array
- Non-integer arithmetic
- Arithmetic other than +, −, *, /
- Multi-letter variable names
- Loops with STEPs other than one
- Boolean operators like AND and OR
- Any BASIC statement containing:

TAB	CONT
CLEAR	LIST
SAVE	LPRINT
LOAD	LLIST
COPY	RUN
NEW	

M-Coder has been available in the U.S. from Kopak Creations of New Jersey. According to Robert Schiller of Kopak, they are discontinuing M-Coder in favor of Z99, a new compiler with increased capabilities. -Ed.

WORD PROCESSING
For ZX81/Timex 1000 with 16K 32K 48K or 64K RAM.
(Please specifiy RAM version.)
ONLY $7.95 (Limited time only)

ALSO AVAILABLE:
Checkbook Balancer—helps with the monthly chore! Ahamur—are you able to be King? Rule an island to see! Escape—inside a deadly barrier with you are Xroids. Can you guess who's for dinner? Rebel Raider—Dark Faker is coming! You must stop him! Marketing—buy/sell stocks to make your fortune, or nothing! SPECIAL: Any one program for $7.95 add only $2. for each additional program!

MARANATHA
Box 759-D, Mableton, Ga. 30059

TS 1000 $59.95
You must buy 3 software from us.

UTILITY
Quick Load $11.99
Assembler 11.99
Disassembler 11.99
Compiler 9.95
Graph 9.95

GAMES
Packman $9.95
Snakebite 9.95
Chess 9.95
Fungaloids 9.95

BUSINESS
Bookkeeping $12.99
Financial Analyzer 11.99
Inventory Control 11.99

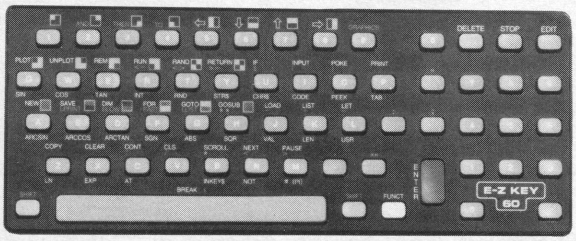
MANY MORE EXCITING PROGRAMS!
CALL FOR FREE CATALOG
DISCOUNT SOFTWARE
320 E. 59th St., NY, NY 10022
Tel: (212) 486-0980

Join the CLICK!

Enter your programs Faster & Easier with the E-Z Keyboard . . .

At last, a large 60 key "Tactile Feel" keyboard that plugs into the same connectors as the existing keyboard on your ZX81 or your Timex-Sinclair 1000.

Hear the CLICK and feel a SNAP for every key pressed! (Tactile Feedback).



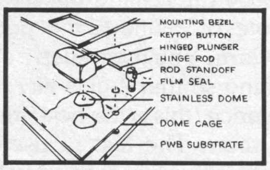
Only $84.95

A custom made enclosure (shown above) is also available for your computer and E-Z Key 60 keyboard.

Measurements:	Price:
EC-11 11"x9"x3"	$25.00
EC-14 14"x9"x3"	$30.00

E-Z Key 60 has the following features:
- 60 Keys - Legends in 3 colors on the base
- Molded legends on key tops
- 8 Automatic shift keys (no shifting required) for edit, delite, single and double quotes, colon, semi-colon, function and stop.
- 2 Shift keys - **Numeric key pad** - 5" space bar.

E-Z Key 60 requires no wiring (just plug it in) and can be adapted to fit the XZ80 or the MicroAce (8 KROM). The Mounting base measures 10"x4". Cables and instructions are included.



SWITCH SPECIFICATIONS:
Key tops measure .4" x .3" - spaced at ¾" intervals between keys. Life = 10 million operations, typical. Force = 3. oz. Travel = .040". Dome switch, button type with arm to give .040" travel.

WATCH FOR THESE NEW PRODUCTS!

JOYSTICK: Joystick kit that requires no wiring and will fuction like the arrow keys & Ø on your computer.

E-Z Key 40 Replacement flat keyboard with embosing around each switch and 3 color legends and graphics as existing keyboard (plug in replacement).

In Stock. 90 day warranty.

E-Z KEY

SUITE 75 A, 711 SOUTHERN ARTERY
QUINCY, MASSACHUSETTS 02169
(617) 773-1187

USE THIS ORDER FORM...

☐ Check or Money Order

Charge to my:
☐ Visa ☐ Mastercard

Card # ________

Expires ________

	Quantity	Unit Price	Total
E-Z Key 60		$84.95	
EC-11/14		$25/$30	
Total units S&H $4 per unit			
Mass, res. add 5% sales tax			
		Total	

Send to:
E-Z KEY
Suite 75 A
711 Southern Artery
Quincy, MA 02169

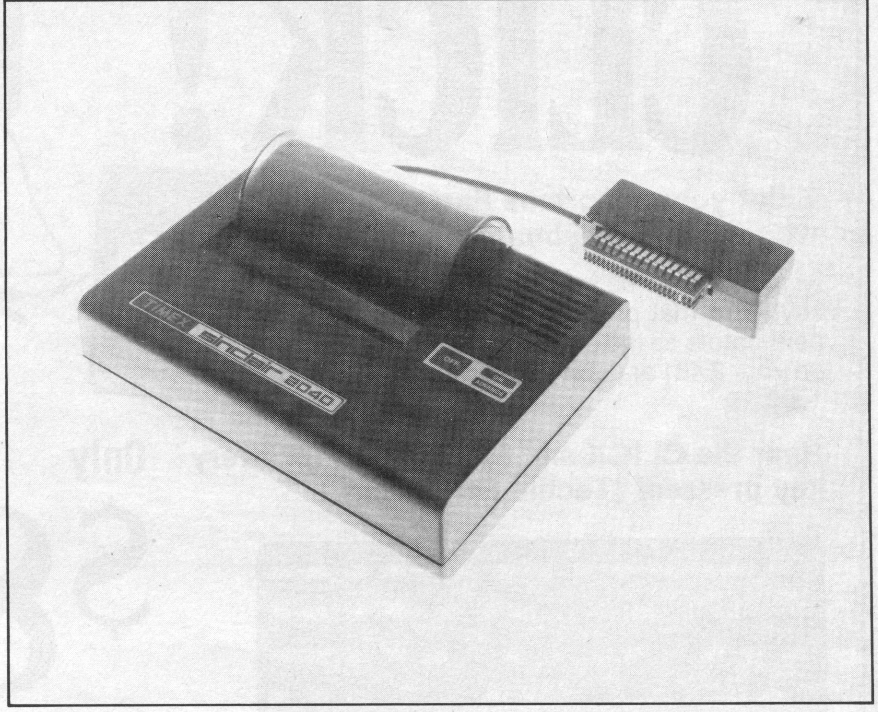
Signature ________

Name ________

Address ________

City ______ State ______ Zip ______

Exclusive Review: Timex Printer



Product: Timex Sinclair 2040
Printer
From: Timex Computer
Corporation
Waterbury, CT
Available in retail stores
Price: $99.95

At last a printer from Timex! The daily trips to my local distributor finally paid off.

Inside the box you find the printer, power supply, paper and instruction booklet. The instructions are brief and to the point. Printer operation is quite simple.

You plug the printer into the expansion port and then your RAM plugs into the printer cable's connector. Timex added a ground strip that also serves to hold the connector to the computer more securely. This addition also stops some of the RAM pack wobble. Only two buttons operate the printer, an off button and a paper advance/on button.

You can check whether everything is OK with the printer's built-in self-test. You hold the on button and press the off button once. The printer then prints a row of the number 8 then a row of the number 1 and repeats until you press the off button again. If your test fails, check your connections and try again. The case contains no user-serviceable parts. I opened it up to check this out. Unless you are highly skilled in electronics, you can do nothing but return it to Timex's service center in Little Rock, AR, for repair.

Ready for disappointment number one? Although the printer comes with one roll of paper, no extra paper came with the distributor's shipment. So use what you have wisely as it may be some time before you get more. (You could cut Radio Shack's thermal paper it to fit, but this is not as desirable.) When loading a new roll of paper, cut the leading edge on an angle and it will feed easier.

OK, disappointment number two. The cable from the printer to the connector that fits the computer's rear expansion port is only six inches long. I have a full-size keyboard and had to open it up and set the printer inside to plug it in. Not very convenient to operate—this may be a problem for some of you.

Operation

Two plates, each containing eight sets of print pins, perform the printing. These pins burn the paper to give the dot pattern impression so that parts of 16 characters print at the same time. The plate moves left to right only about 0.2 inches. Its speed is really good for its size.

Copying the entire screen takes about 11 seconds. The controlling factor is the paper advance speed and not the number of characters on the line. Paper advance takes about 1/2 second per line. To put this in perspective with a full-size printer (which prints only one character at a time like a typewriter), I compared it to my Digital Decwriter IV. Printing a program of 20 lines with 5 characters per line took the Digitial 5.5 seconds. It took almost 24 seconds to list another program of 20 lines with 32 characters per line. My Timex printer required just over 10 seconds to print each of these same programs.

I found the characters easy to read and the print clear. (See Figure 1.) The 2040 uses a 6x6 dot pattern for characters and an 8x8 pattern for graphics and inverse video. It prints ten characters to the horizontal inch and about 22 lines every 2.7 vertical inches (a little over eight lines to the inch). The printout is 32 characters wide, exactly duplicating the screen.

One note of caution: the printer does get warm. You must keep it clear of everything so that it gets air.

For the price, this is the best printer I've ever seen. The test of use will determine if it is really a good value. Timex does not furnish any information about how long their 2040 works before needing repair. Timex offers a 90-day limited warranty. (Users report that the TS2040 does not work with Memotech 32K RAMs. It works fine with 16K and 64K RAMs. Memotech confirms this finding and is working to learn why.—Ed.)

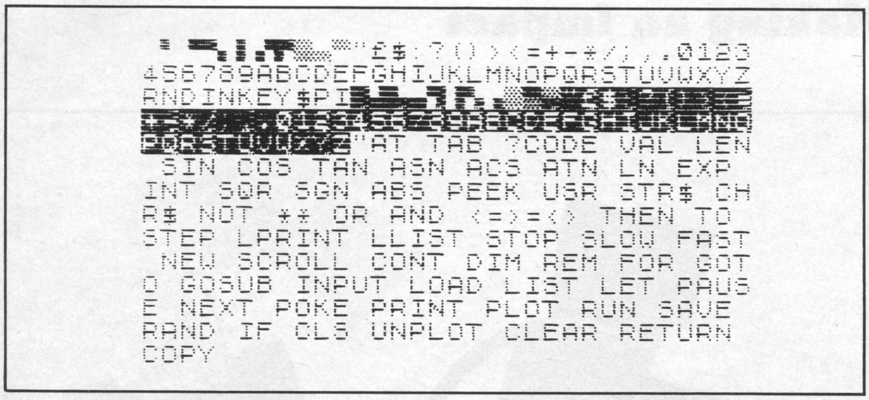


Figure 1.

Specifications

Interface: None required with ZX81 or TS1000 computers.

Material: Impact-resistant ABS plastic.

Size: Printer—7 3/4 x 5 5/8 x 3 1/4 inches. Power supply—2 3/4 x 3 1/4 x 2 1/4 inches.

Power: 120V AC, 60 cycle, 35 Watt input. 24V AC, 1.2 Amp output.

Paper: Standard thermographic printer paper, roll width 110 mm or 4 1/3 inches, roll diameter 48 mm or 1.9 inches, paper length 25 m or 82 feet maximum. Japan Paper and Pulp Co. type TP50CM-A (blue) and TP50KM-A (black). Available through Timex distributors.

Dale Lipinski
Roslyn, PA

SLOW Motion

Product: Video Upgrade Board Kit
From: Computer Engineering Services
P.O. Box 1222
Show Low, AZ 95901
Price: Full kit $32.50 + $2.50 P&H
PCB only $19.95 + $1.25 P&H
Parts only $12.55 + $1.25 P&H

Computer Engineering Service's video upgrade board is a do-it-yourself hardware project that allows your ZX80 with 8K ROM or similar MicroAce computer to operate in SLOW mode. (Upgrading a 4K ROM ZX80 with a replacement 8K ROM provides only 8K functions. SLOW mode is implemented in ZX81s and TS1000s in hardware.—Ed.) This kit is the only product presently available in the U.S. to make this modification.

With a full kit you get a PC board, parts package, documentation with assembly instructions and schematic. You can also get a partial kit. The PC board has printed traces on both sides but does NOT have plated-through holes, causing several problems later in assembly. I found the instructions quite clear. They spell out separate upgrades for ZX80s and MicroAces. The schematic is somewhat crude, being a copy of a rough hand-drawn original. The wire supplied to connect the video board to the main board is 30-gauge wire-wrap.

Assembly went reasonably well and took me about two hours. Several PC board traces pass rather close together so you must take care in soldering to prevent bridging. You solder the ICs directly to the PC board. One of the final assembly steps is to solder some feed-through jumpers to connect top and bottom traces together. One feed-through was hard up against an IC, between the IC and a small capacitor, and difficult to reach with the soldering iron.

I have assembled mumberous kits, but they always came with plated-through holes so that I needed to solder on only one side of the PC board. My video board would initially not function and I found I indeed forgot to solder several resistors on the top side.

You must make 14 connections between the video board and the ZX80 board. The 30-gauge connecting wire proved somewhat fragile and gave some solder joint proplems. I recommend substituting heavier gauge stranded wire for the connections, in case any movement occurs between the boards during final case installation.

According to the instructions you can mount the board in the original case. My previous hardware modifications and additions to the case prevented even attempting internal mounting. Even so, it looks like a tight fit, and I seriously doubt you can mount the board without risking inadvertent contacts and shorts or breaking the fragile connector wire.

My kit does work and the results are quite pleasing. The top line of the display has only a very slight slant apparent only on close scrutiny. If you have an 8K ZX80 this is the only way to get the important SLOW mode that allows continuous display.

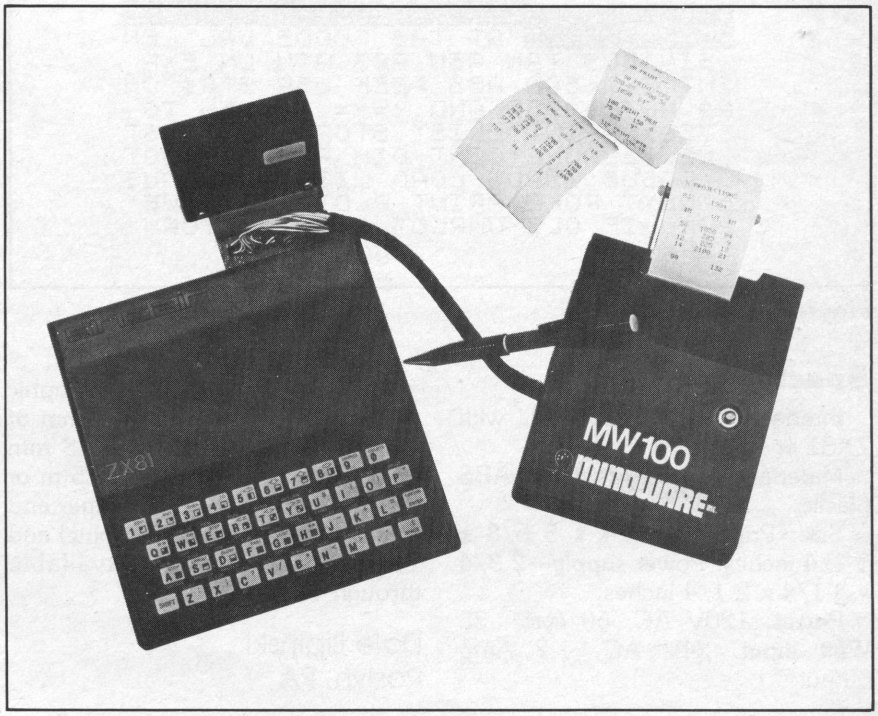
Peter D. Hoffman
Corpus Christi, TX

Get the Static Out
UHF MODULATOR
$15. + SHIPPING
Switch to Ch 35. Clears TV interference. Replaces VHF unit inside ZX/TS case. Requires soldering skills.
Stop The Crashes
RIBBON CABLE ASSEMBLY W/8" cable.
$18. + SHIPPING
We also assemble connectors to your specs. Connectors are available separately for $7 each.
OTHER PRODUCTS: Buffered Buss Kit $65. 8 ch A/D+8 ch D/A Board $195. Storage scope program $25. Fast Fourier Transform program. More.
Quick Delivery. FREE Catalog
$3 shipping charge. Calif. res. please add 6½% tax. To order send check or money order or call for C.O.D.
Computer Continuum
301-16th Ave
San Francisco, CA. 94118 (415)752 6294

SQ HARDWARE REVIEW

Making an Impact

Product: MW-100 Printer
Mindware Inc.
15 Tech Circle
Natick, MA 01760
617/655-3388
Price: $119.95 (plus $4.95 P&H)
Also available in retail stores at $119.95.



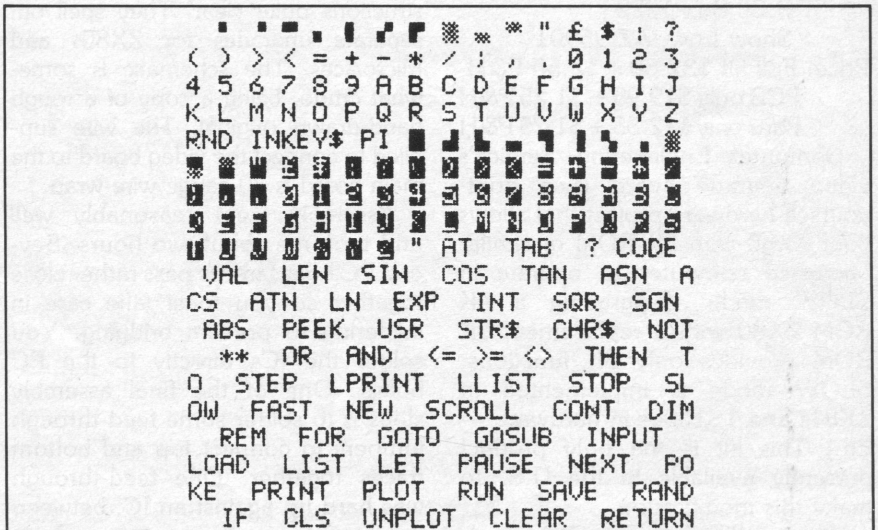
Mindware's MW-100 printer is the first ZX/TS printer available in the U.S. both by mail order and in retail stores. For $119.95, you get a printer, one roll of paper, a power supply and a 13-page instruction manual. The MW-100 is a 16-column, 5x7 dot matrix impact printer, printing approximately one 5x7 dot matrix line per second. It uses 1 3/4 inch plain paper and a printer ribbon like Radio Shack's Pocket Computer printer.

This printer is a compact, attractive package with a stiff 6-inch cable that plugs into the edge connector on the computer rear. You can piggy-back external RAMs into the printer connector. The power supply furnished with the printer powers the computer-printer combination, or you can use the 1.0 Amp power supply furnished with early TS1000s. Sinclair's or Timex's 650 mA power supply will not power this printer.

You choose among three print modes selectable by POKE statements in immediate mode or in program. Mode 0 prints with wrap-around (16 columns, so one 32-column line prints as two 16-column lines). Mode 1 splits the screen into 16 column format and prints left half first then the right half. Mode 3 prints the left half only. Mode 2 does not now exist. According to Mindware, they abandoned whatever they planned for Mode 2 in favor of a more complete redesign in the future. Sinclair's keyboard printer commands, LLIST, LPRINT and COPY, operate the MW-100. The BREAK key terminates printing in any function.

Typical of impact printers, the unit is relatively noisy but gives good print quality that I find easy to read. Printed characters such as "u" and "w" are easy to distinguish compared to some printers' output.

According to the manual, refill paper is available from office supply stores. I found that it is not easy to locate. Mindware will furnish the names and phone numbers of sources. Once I learned that it is cash register paper, I easily located it at a cash register dealer but in rolls too



ZX/TS character set printed by MW-100 printer in mode 1 (split-screen).

large in diameter to fit on the printer paper holder. Because the printer carries the paper roll externally behind the printer, you could easily construct a holder for the larger roll. In addition, the larger rolls are inexpensive — $2.50 for five rolls of 3-inch diameter.

I found, contrary to what the manual states, that any mode other than 0 must be POKEd *after* a program is LOADed instead of before. A call to Mindware confirmed that this was an error in the instruction manual. I also learned (not in the manual) that LPRINT will not work in split screen format (Mode 1). LPRINT, LLIST, and COPY all work in modes 0 and 3.

Overall, I am pleased with the printer. I would prefer a 32-column format but I like the inexpensive plain paper and the ability to use keyboard printing commands. My two basic complaints are that I wish Mindware had provided for the larger diameter paper commonly available and that the connecting cable were less stiff to allow the printer to sit closer to the computer instead of straight out to the right side.

Russell Crum
Canton, MI

SINCLAIR / REAL TIME CLOCK ZX80/1 TIMEX 1000

I/O Board with our Real Time Clock/Calendar

Time	Month	Date	Year	Day of Week
23:59:59	12	31	99	7

Features
- 8 Outputs capable of driving relays
- 8 TTL inputs
- Feed through Sinclair Bus connector to allow normal expansion
- Battery back-up for clock
- Expandable Ports
- All software included
- 90 day warranty

Future Products
- Touch Tone™ encoder/decoder
- Speech and Sound synthesizer

MODEL Built & Tested	MODEL Bare PCB & Manual	DESCRIPTION	PRICE
310		MASTER I/O WITH CLOCK	$59.95
	315		$24.95
MODEL 310 ACCESSORY BOARDS			
320		A to D and D to A converter	$47.95
	325		$19.95
330		Wireless control system (BSR™)	$69.95
340		Solid State AC Relay	$19.95
SPECIALIZED PARTS FOR MODEL 315			
316		Sinclair edge connector, 46 pin w/key	$ 4.95
317		Clock IC and crystal (tested)	$12.95

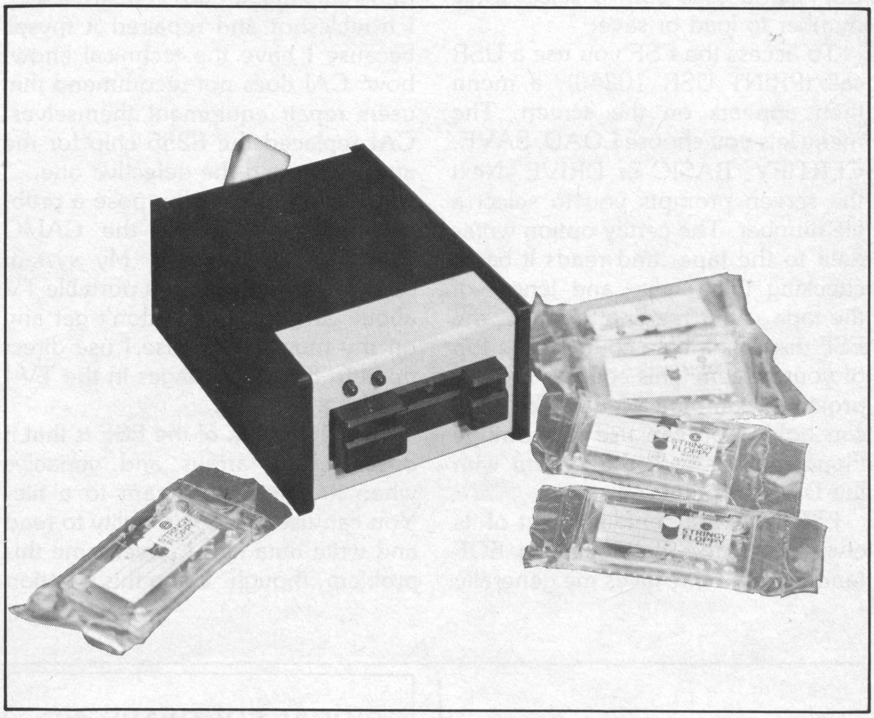
California residents add 6% Shipping and handling $ 3.95

Send self addressed stamped envelope for catalog.

VISA/MASTERCARD

JK AUDIO
P.O. Box 3295
Escondido, CA.92025-0580
(619) 741-5132 (24 Hr. Order Line)

The Floppy Alternative



Product: CAI Stringy Floppy
From: CAI Instruments
P.O. Box 2032
Midland, MI 48640
517/687-7343
Price: $109.95 Stringy Floppy
$69.95 CAI/O Board

Although not quite as fast as a floppy disk, CAI's Stringy Floppy is an inexpensive an alternative to a disk drive. After using the Stringy Floppy, you'll never want to use a cassette again.

Exatron of California manufactures this fast, quiet, low cost mass storage device. Physically small, it measures only 4x6x2 1/2 inches.

To use CAI's Exatron Stringy Floppy (ESF) you need the CAI/O board interface, which plugs into the rear of the ZX/TS. The CAI/O board contains the firmware to operate the ESF. With the CAI/O board you also get a serial RS232 I/O port, and three parallel I/O ports, as well as a port for the CAI/P40 printer and one for a 16K or larger RAM pack.

Instead of cassettes, the ESF uses tiny endless-loop tapes called wafers. CAI supplies these wafers in 5-, 10-, 20-, 35-, and 50-foot lengths. Five-foot wafers hold approximately 8K bytes while 50-foot wafers hold approximately 80K bytes. Wafer prices range from $2.95 for 5 feet to $4.95 for 50 feet.

My ESF is very fast, inputting and outputting data at 14000 baud, about 56 times faster than a cassette recorder. A 16K program loads in approximately 10-12 seconds. If you position the tape just after the program you want to load, it takes about 20-30 seconds longer, depending on tape length, because the tape must advance to the beginning of the selected file. Saving a program takes about twice as long as loading because the ESF reads what it has just written to check for the correct save. (John Bauriedel from CAI says ESF's baud rate is between 11000 and 14000 baud, or about 1K of memory per second. He points out that this rate is *faster* than the transfer rate reported for Macronic's disk drive in SQ, Spring 83—Ed.)

Programs are saved in files. The ESF menu asks you to select a file number to load or save.

To access the ESF you use a USR call (PRINT USR 10240); a menu then appears on the screen. The menu lets you choose LOAD, SAVE, CERTIFY, BASIC or DRIVE. Next the screen prompts you to select a file number. The certify option writes data to the tape, and reads it back, checking the quality and length of the tape. After reading the tape, the ESF displays a byte count at the top of your screen. This count is the approximate amount of data the tape can hold. You can use two Stringy Floppy Drives with the system with the DRIVE option.

ESF firmware contains a set of its own error messages, such as EOF (end of file). Mine gives me generally reliable operation, with an occasional error message.

I had one major malfunction—the 8255 chip in the CAI/O board failed. I troubleshot and repaired it myself because I have the technical know-how. CAI does not recommend that users repair equipment themselves. CAI replaced the 8255 chip for me after I returned the defective one.

TV interference may pose a problem for you because the CAI/O board is not shielded. My system causes interference on a portable TV about 20 feet away. I don't get any on my monitor because I use direct drive to the video stages in the TV I use for a monitor.

One drawback of the ESF is that it doesn't save arrays and variables when it saves a program to a file. You can use the ESF's ability to read and write data files to overcome this problem, though. Using this function requires that you include subroutines in your program to accomplish this. The manual gives examples.

Another disadvantage is that you can't make your program start automatically after loading as with a cassette. Although ESF provides a feature called program chaining, where you can load one program that loads and runs another, it's not the same.

A list of assembly language routines is in the manual for those with knowledge of assembly language.

Instruction booklets accompanying these units contain schematics. Both the ESF and the CAI/O board come with a 30-day warranty.

Richard Graffius
N. Versailles, PA

XFORTH©

WRITTEN IN ENGLAND BY PROFESSIONAL PROGRAMMERS

- Sinclair ZX-81 "Timex T/S 1000"
- Very full FORTH-79 sub-set
- On cassette for 16K (or more)
- Loads in one pass (unlike some versions which require up to four extra editor loads)
- Compact coding - 10K+ still available for user code.
- Sinclair specific look up tables (uses Sinclair code instead of ASCII)
- Extra utilities - CLS, PLOT ...
- BASIC call - return to BASIC with simple, one word inst.
- Only $25.00/tape + $1.00 P&H. Arkansas residents add $1.00.

Formerly distributed only by ZETA™ SOFTWARE, we have the original **FOOTBALL FORECASTER** with 1983 data base. Available for 16K ZX-81, T/S 1000 or 16K TRS-80 Color Computer. Specify NFL or College. Only $19.95 each or $29.95 for both. Add $1.00 P&H. Ark. residents add 4% Tax.

HAWG WILD SOFTWARE™
P.O. Box 7668
Little Rock, Arkansas 72217

BIOCAL SOFTWARE INC.

ORIGINAL PROGRAMS FOR TS/1000 & ZX81

CHECKBOOK
THE STOCK EXCHANGE
BLACKJACK/BAR-DICE
NIM/TAC-TIX
BIORHYTHM
TYPIT
GIN-RUMMY/YUKON SOLITAIRE
KLINGON WARS
CRIBBAGE

PLUS MANY MORE...

$10.00/ea. Cassette Tape
$15.00/ea. Stringy Floppy

For credit-card orders call us
Toll Free · **800-237-8400/ext. #70**

Write for **FREE CATALOG** or call
(415) 892-1747

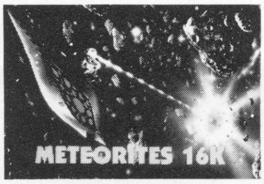
To Order by Mail, send check or money-order

To: BIOCAL SOFTWARE INC.
Dept. R.
340 Cypress Drive
Fairfax, CA 94930

DEALER INQUIRES WELCOME
WE GUARANTEE OUR PRODUCTS

Ad Index

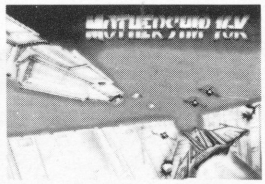
TIMEX SINCLAIR 1000/1500 SOFTWARE
SOFTSYNC, INC.



METEORITES Shoot your way out of the asteroid field.
16K $14.95

RED ALERT Scramble between treacherous mountains and fearsome aliens.
16K $14.95

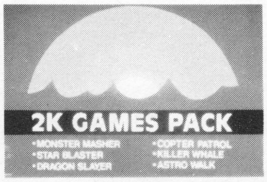
ALIEN INVASION The best TS/ZX version of this classic game.
16K $14.95



MOTHERSHIP Zoom down the Zarway corridor, dodging and shooting drones, to get a clear shot at the ominous mothership. Amazing graphics!
16K $16.95

NIGHTGUNNER Maneuver your tail gun as you fire at enemy aircraft coming at you from all directions. Features on-screen scoring and bonus points.
16K $14.95

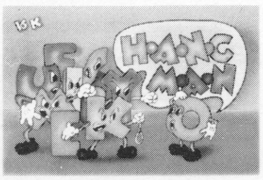
SPACE RAID/ TS DESTROYER Two great fast action 2K programs on one tape.
2K $14.95



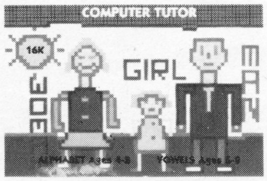
2K GAMES PACK 6 great graphics games on one cassette: COPTER PATROL, KILLER WHALE, ASTROWALK, MONSTER MASHER, STAR BLASTER, DRAGON SLAYER.
2K $14.95

PROGRAMMERS TOOLKIT Machine code routines to help programmers write better software.
16K $14.95

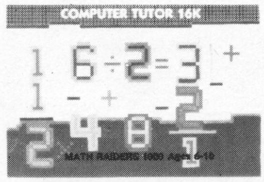
GRAPHICS KIT Machine code routines to help create exciting graphics in programs.
16K $14.95



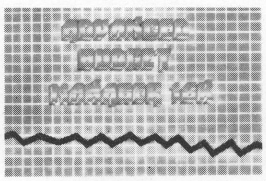
HANGMAN Seven versions of this classic guessing game including a user created library of 500 words.
16K $14.95



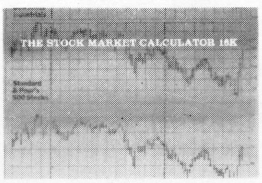
ALPHA/VOWEL TUTORS Two graphic tutorials designed to teach the alphabet as well as long and short vowels.
16K $14.95



MATH RAIDERS Teaches children basic mathematics through the use of a graphic game.
16K $14.95



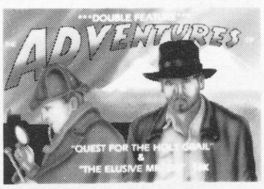
ADVANCED BUDGET MANAGER Stores a year's worth of projected and actual income and expenses.
16K $16.95



THE STOCK MARKET CALCULATOR Two programs to aid the investor in calculating stock and option investments.
16K $16.95

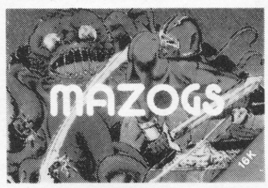
SPACE COMMANDO This adventure takes you to Zircon 12 to overcome the alien forces and liberate the planet.
16K $14.95

SHARK'S TREASURE In this adventure brave the deadly shark infested waters to find the treasure.
16K $14.95

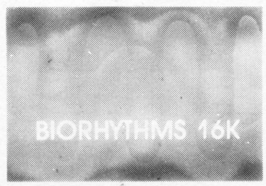


THE ADVENTURES of "QUEST FOR THE HOLY GRAIL" and "THE ELUSIVE MR. BIG" Two different adventures on one tape. Special feature creates new versions every time.
16K $17.95

SUPERCHESS The most advanced chess available featuring ten levels, tactical or positional analysis, self play, and will recommend moves.
16K $19.95



MAZOGS The most elaborate, graphic maze adventure available for the TS/ZX computers.
16K $19.95



BIORHYTHMS Chart your Physical, Emotional and Intellectual ups and downs.
16K $14.95

Write or call for free catalog and program listing.

SOFTSYNC, INC., 14 E. 34TH ST., NEW YORK, N.Y. 10016 (212) 685-2080

DEALER INFORMATION AVAILABLE **MC/VISA ACCEPTED**

KOPAK

Creations Inc.

ACCOUNTS

Your own PERSONAL OR SMALL BUSINESS accounting system. Contains refinements not usually found in this type of program. "200 Transaction file" can be SEARCHED, PRINTED or DISPLAYED. This program has been throughly tested, and improved through extensive use.

ORDER No. 03-1022 $19.95

AUTOFILE

An ELECTRONIC FILING SYSTEM that can be used for ANY type of data. Set up your own DATA-BASES with this easy to use, menu-driven program with AUTO-SEARCH facility.

ORDER No. 03-1021 $19.50

BREAKOUT

Fast MACHINE CODE arcade-type game. If you haven't tried THIS version, you haven't played "BREAKOUT" on the ZX81/TS1000.

ORDER No. 03-4028 $14.95

COMPOSER

Enables you to play any tune through your T.V. speaker, your cassette recorder or through an A.M. radio. With 'COMPOSER' you can control the duration of each note. And you can save your song on tape.

ORDER No. 03-3012 $19.95

G R A F I X

Draw and store complex images. Permits you to create, mix, and save your own screen artwork. You can mix graphics created with GRAFIX into another program. By itself or on tape. This program is a "must" for the CREATIVE PROGRAMMER.

ORDER No. 03-6005 $14.95

Z-99™ (Improved 'M'-Coder)

Now you can write MACHINE CODE PROGRAMS on your ZX81/TS1000 with Z-99. The best true MACHINE CODE COMPILER for the ZX81/TS1000 computer. Simply write your new program in BASIC (or load an EXISTING PROGRAM), press a key and Z-99 will automatically compile it into MACHINE CODE for you. Give all your programs that SUPER—FAST professional look. Order Z-99 NOW to revolutionize your programming. Once you try Z-99 you'll never write basic programs again!

ORDER No. 03-5271 $19.95

PUCKMAN

Superb fast moving MACHINE-CODE fun. Have an exciting time playing this EXCELLENT version of the popular arcade game.

ORDER No. 03-4027 $14.95

'TOUCH-A-MATIC'™

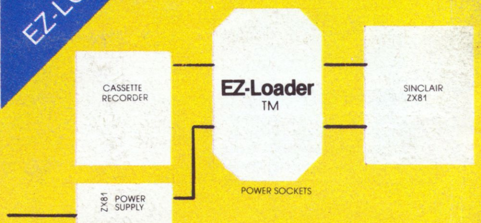
$9.95

POWER AT YOUR FINGER TIPS!

ORDER No. 01-0007

KOPAK'S "TOUCH-A-MATIC" gives you the power to type more accurately and faster. It requires NO WIRES, NO SOLDERING. It's as easy as removing the adhesive backing and pressing it into position.

EZ-LOADER™



LOADS & SAVES 16 K file in 26 seconds instead of 7 minutes! Cleans up signal for a cleaner load. • STRONGER, SHARPER DIGITAL SIGNAL • CLEANER, FILTERED POWER • ENCODES YOUR PROGRAM AND/OR DATE • Save time/Save energy • Baud rate increases from 250 to over 4000 bps. A must for all ZX81/TS1000 owners.

ORDER No. 04-1010 $39.95

Dealer Inquiries Invited

ORDER FORM

TO AVOID DELAY PRINT CLEARLY

MAIL THIS FORM WITH PAYMENT TO:
KOPAK Creations Inc.
119 Peter St.
Union City, NJ 07087
(201) 864-4410

STOCK NUMBER AND DESCRIPTION	HOW MANY	TOTAL
	Sub-Total	$
	Add sales tax	$
	Packing & Shipping (See Chart)	$
	Total amount enclosed or to be charged to my credit card	$

NAME ____
ADDRESS ____
CITY STATE ZIP

SHIPPING
up to $9.99 to $14.99 add $2.00
from $15.00 to $19.99 add $2.50
from $20.00 to $24.99 add $2.85
from $25.00 to $29.99 add $3.45
from $30.00 to $39.99 add $3.70
from $40.00 to $49.99 add $3.95
over $50.00 add $4.25

CHARGE
VISA ☐
MASTER CARD ☐
INTERBANK NO.
EXP. DATE MO YEAR
Sign ____

In Canada call Datronics (416) 496-2243